For Tandy's 100 and 200 Portables and 1000, 1200 and 2000 MS-DOS Computers

PCM®

The Personal Computing Magazine for Tandy® Computer Users

Vol. III No. 4
October 1985
U.S. $3
Canada $3.95

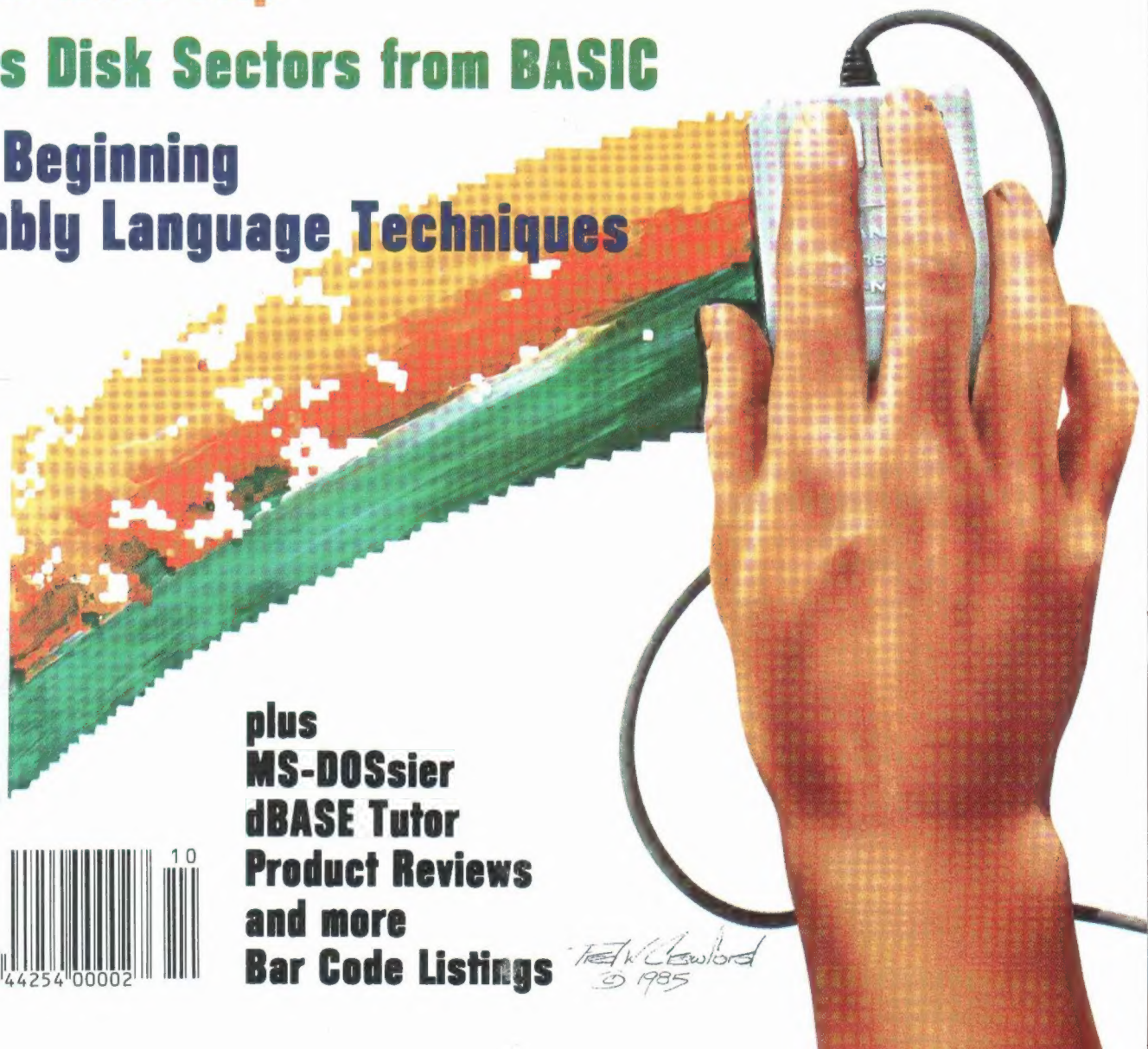
Sketch

Turn Your Screen Into a Canvas and Your Mouse Into a Paintbrush

Take a Portable Navigator on Your Next Trip

Access Disk Sectors from BASIC

Learn Beginning Assembly Language Techniques



plus
MS-DOSsier
dBASE Tutor
Product Reviews
and more
Bar Code Listings

SIMPLE SOLUTIONS

TELECOMMUTER™

INTEGRATED WORD PROCESSING AND COMMUNICATIONS SOFTWARE

For the Tandy 1000, 1200 and 2000, and IBM-PC/XT/AT and compatible computers. Uses all the Model 100/Tandy 200 TEXT and TELCOM commands.

WHY BUY JUST FILE TRANSFER? TURN YOUR DESKTOP COMPUTER INTO A 'SUPER MODEL 100/TANDY 200'!

For $200 you can put all this on your desktop computer:

- Full function Word Processor based on and compatible with Model 100/Tandy 200 TEXT Editor.
- Super Power Communications based on and compatible with Model 100/Tandy 200 TELCOM.
- Fast File Transfer between Model 100/Tandy 200 and desktop at 9600 baud. Fully prompted at every step so you'll never get lost. And we supply the cable at no extra cost!
- Host Mode for remote control of desktop computer from Model 100, Tandy 200, dumb terminal, or any computer with Telcom capability.

You don't have to put extra code on your Model 100 or Tandy 200 to transfer files — **no cassettes, no ROM cartridges. Telecommuter** will exchange files with any computer that has an RS-232 port and communications software, even another desktop computer.

Optional advanced features include XMODEM protocol file transfer, full VT100 terminal emulation, Telcom scripts and macros for automatic dialup and file transfer, DOS access from editor, and multiple access level Host Mode. **TC-Corporate** allows individual user assignments, such as password, access level and time limit; keeps activity log.

Telecommuter is a new and significantly easier way to use your portable and desktop computers together. Call or write today for more information on our entire line of fine software products.

We accept MasterCard and VISA for mail or telephone orders, and we ship free in the USA. Upgrade any time for price difference.

TC-standard or **TC-XMODEM** $200, **TC-DeLuxe** $300, **TC-plus** $400, and **TC-Corporate** $1000.

Look for Telecommuter at your local Radio Shack Computer Store!

SIGEA SYSTEMS, INC.

19 Pelham Road, Weston, MA 02193 (617) 647-1099

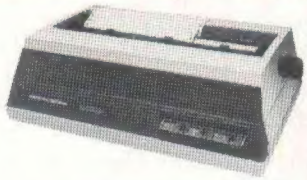
From Computer Plus to YOU . . .

PLUS after PLUS after PLUS

Tandy 200 24K $739
Model 100 24K $510

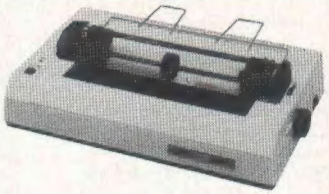
Tandy 1200 HD $1599
10 Meg. 256K

Tandy 1000 $779
Tandy 1000 HD $1539



Smith Corona L-1000
$259

Color Computer II
w/64K Ext. Basic $179



DMP105 $160

BIG SAVINGS ON A FULL COMPLEMENT OF RADIO SHACK COMPUTER PRODUCTS

COMPUTERS	
Tandy 1000 1 Drive 128K	779.00
Tandy 1000 HD 10 Meg. 256K	1539.00
Tandy 1200 10 Meg. 256K	1599.00
Tandy 2000 2 Drive 256K	1295.00
Tandy 2000 10 Meg. 256K	2025.00
Model IVD 64K with Deskmate	889.00

PRINTERS	
Radio Shack DMP-130	269.00
Radio Shack DMP-430	660.00
Radio Shack DWP-220 Daisy Wheel	469.00
Radio Shack TRP-100 Portable	229.00
Silver Reed EXP-550 Daisy Wheel	425.00
Star SG-10	245.00
Star SG-15	410.00
Star SD-10	365.00
Panasonic P-1091	275.00
Toshiba 351	1175.00
CITOH Prowriter 8510AP + NLQ	345.00
Okidata 192	375.00
Epson LX-80	245.00
Epson FX-185	369.00
Epson HS-80 Portable Ink Jet	339.00

Tandy 1000/1200 ACCESSORIES	
Tandy 1000 Disk Drive Kit	159.00
Tandy 1000 10 Meg. Hard Drive	579.00
Hard Drive Controller Board	265.00
256K Ram Board (inc. 128K & DMA)	199.00
512K Ram Board (includes 128K)	169.00
256K Memory Plus Expansion Brd.	249.00
PBJ Multi-Function Board (128K)	269.00
PBJ Multi-Function Board (256K)	299.00
PBJ Multi-Function Board (512K)	349.00
128K Ram Upgrade Kit (NEC)	78.00
256K Ram Upgrade (for PBJ Board)	69.00
300 Baud Modem Board	129.00
1200 Baud Modem Board	249.00
RS-232 Serial Board	89.00
Digi-Mouse/Clock Board	89.00

Tandy 1000/1200 MONITORS	
Tandy VM-2 Green Monitor	129.00
Tandy CM-2 RGB Color Monitor	379.00
Tandy VM-3 TTL Green Monitor	179.00
Tatung CM-1360 Color/Gr./Amber	459.00
Tandy 1200 Text Monitor Adapt.	179.00
Tandy 1200 Color/Graphics Adapt	240.00

MODEMS	
Radio Shack DCM-3 Modem	52.00
Radio Shack DCM-5 Modem	99.00
Radio Shack DC Modem 2212	315.00
Hayes Smartmodem II 300 Baud	169.00
Hayes Smartmodem 1200 Baud	429.00
Novation J-Cat 300 Baud	115.00

FOR EVERYONE	
Network Four Outlet Surge Prot.	69.95
CCR-81 Cassette Recorder	52.00
CCR-82 Cassette Recorder	43.00
C-20 Digital Cassette Tapes (10pk)	9.95
Verbatim SSDD Plastic Box (10pk)	24.95
Verbatim DDDD Plastic Box (10pk)	29.95
Generic SSDD Diskettes (10pk)	17.95
NEC 64K Ram Chips (set of 8)	39.00
Radio Shack 16K Ram Chips (8)	25.00
Locking Disk Box (holds 70)	19.95
Flip N File Disk 50	15.00
Fanfold CleanEdge Paper (2600)	35.00

Radio Shack software 10% off.
Send for complete listing of
brand name software and hardware.

CALL TOLL FREE
1-800-343-8124

- *LOWEST POSSIBLE PRICES*
- *BEST POSSIBLE WARRANTY*
- *KNOWLEDGEABLE SALES STAFF*
- *TIMELY DELIVERY*
- *SHOPPING CONVENIENCE*

VISA · AMERICAN EXPRESS · MasterCard

computer plus

P.O. Box 1094
480 King Street
Littleton, MA 01460

SINCE 1973

IN MASSACHUSETTS CALL (617) 486-3193

TRS-80 is a registered trademark of Tandy Corp.

PCM

The Personal Computing Magazine for Tandy® Computer Users

October 1985 Vol. III No. 4

PCM — The Personal Computing Magazine for Tandy® Computer Users is published every month of the year by FALSOFT, INC., The Falsoft Building, P.O. Box 385, Prospect, KY, 40059. Phone (502) 228-4492. **PCM — The Personal Computing Magazine for Tandy® Computer Users** and the **PCM** logotypes are registered ® trademarks of FALSOFT, Inc.

Second class postage paid Prospect, KY, and additional offices. USPS N 713-470 (ISSN 0747-0460). POSTMASTER: Send address changes to PCM, P.O. Box 385, Prospect, KY 40059. Forwarding Postage Guaranteed.

Subscriptions to **PCM — The Personal Computing Magazine for Tandy® Computer Users** are $28 per year in the United States. Canadian rates are U.S. $35. Surface mail to other countries is U.S. $64, air mail U.S. $85. All subscriptions begin with the next available issue.

Payment accepted by VISA, MasterCard, American Express, Cash, Check or Money Order in United States currency only.

Limited back issues are available. Please see notice for issues which are in print and costs. Payment accepted by VISA, MasterCard, American Express, Cash, Check or Money Order in U.S. currency only.

Contents

Features

Departments

Reviews

Tandy computers offer everything ...almost

Check out the software that completes their systems —*filePro*®

Tandy MS-DOS Software Comparison Chart

	pfs® File/Report	dBase™ II	*filePro* 16 Smallware
GENERAL CHARACTERISTICS:			
Menu driven	yes	no	YES ✓
Allows user to create integrated business systems	no	programmer required	YES ✓
Developed systems and data can be moved to multi-user environments	no	no	YES ✓
Professional support available from the software's authors	no	no	YES ✓
PRICE	$265	$595	$495
CAPACITIES:			
Fields per record	100	32	999 ✓
Characters per record	1679	1000	4608 ✓
Records per file	1300	65535	16,000,000 ✓
Indexes per file	1	7	12 ✓
Number of digits per numeric field	20	10	24 ✓
Number of files usable concurrently	1	2	10 ✓
Files span multiple drives	no	no	up to 8 ✓
FEATURES:			
Full-screen facility for creating custom screen layouts	yes	no	YES ✓
Full-screen facility for creating custom report layouts	no	no	YES ✓
Built-in field types (error checking)	no	3	12 ✓
User-defined field types	no	programmer required	200 ✓
Conditional math	no	programmer required	YES ✓
User-defined menus	no	programmer required	YES ✓
Change file layout without losing existing data	possible	possible	automatic ✓
Data protection	no	programmer required	YES ✓
Password security	no	programmer required	YES ✓

The new Tandy computers can really speed things up. If you don't let obsolete software bog you down.

Try *filePro* 16, the latest generation of data base software from the authors of Profile®.

Choose it for its power (check our capacities); its ease-of-use (check our features); and its unique ability to grow as your needs grow, and even move the systems you create to multi-user computers.

filePro is available for the Tandy 1000, 1200 and 2000 computers (hard disk required). Available directly from us or through your Radio Shack store's Express Order Software service.

To order, call (800) 847-4740.

For customer service or when dialing in New York State, call **(212) 398-9290.** Ask for Smallware sales. Visa and MasterCard accepted.

Also available for other computers, including IBM PC, IBM PC-XT, and NCR Decision Mate.

The Small Computer Company, Inc.

✓240 **230 West 41st Street, Suite 1200, New York, New York 10036**

A Word to the Doomsayers

I dislike doomsayers and I suppose this summer will someday be known as the summer of the personal computer industry's discontent.

Just about everywhere, even in some publications which contend they *support* a particular computer, there have been stories aplenty about how the computer industry is in a slump, how sales are off and, in some cases, a wondering aloud about whether a computer is all it's cracked up to be.

There are some real good reasons for the doomsaying, but there are many solutions as well. The solution would be obvious to anyone who might have caught a certain full-page advertisement in the *Wall Street Journal* last week. The ad was placed by Tandy Corp.

The advertisement compared the IBM PC, XT and AT to the Tandy 1000, 1200 and 2000. Much of the comparison was on the basis of price, but there were other comparisons too — based on the sheer quality of the hardware. Yes, I know, this was all written by the people in Fort Worth, but nevertheless, the day is long since past when you can put anything but the truth in an advertisement.

The truth is that the Tandy MS-DOS family comes out head and shoulders above the IBM family. In terms of abilities; in terms of support; in terms of cost. Pure and simple.

So why do we have so much doom and gloom?

I think I know the reason. Most of the "popular press" have, for quite some time, been enamored by two "stars" — the IBM PC and the Apple MacIntosh. But, in recent years and recent months, I believe the average consumer has become more aware of the limitations of those machines and they have been searching around for something else. They have put off buying to see what develops.

If *you* know someone like that, tell him or her the search is over. Just lead them to a Radio Shack store or Computer Center.

If you wish ease of use, get a 1000 or 1200 and use GEM. Same excellent mouse-supported friendliness as the Mac — and at a lot greater speed. If you want to number crunch, try the Tandy 2000 — my "*Norton Utilities* Index" tells me the 2000 is several times faster than the IBM PC. And, if you *also* want some software (which you do), you can get the computer *and* some software from Tandy for the same price as the hardware alone from IBM or Apple.

Stewart Alsop wrote recently that he has a theory of how computer sales will go. He said that when everyone expects sales to be good, they are usually bad; and vice versa. Since everyone expects the last quarter of the year to be a dog, Alsop says he thinks it will be truly excellent.

I agree. But I happen to think that is because people will be finding the power and the price equation from Tandy much too difficult to resist. And I think this realization of power and price will translate into an excellent fourth quarter computer market for those who can compete.

I know Tandy can and will.

I'd be remiss if I did not just add a little note that PCMfest in Princeton, N.J. is almost upon us. It is Oct. 11-13 and we certainly hope you will be with us.

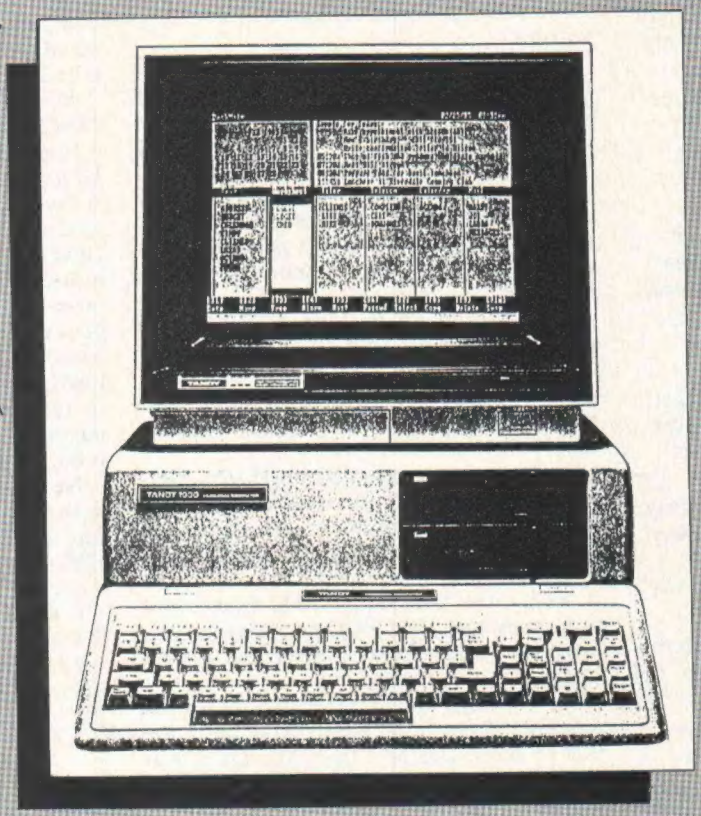
Check out the official announcement elsewhere in this issue of PCM and join us in Princeton! □

— Lonnie Falk

TANDY 1000
INTERNAL HARD DRIVES

- NO ADDED MEMORY BOARD NECESSARY
- BOOTS OFF HARD DRIVE
- TERMS—
  MONEY ORDER
  CASHIER'S CHECK
  *C.O.D.
  *MASTERCARD/VISA
  5% FEE

*PLUS SHIPPING & HANDLING



- COMPLETE KIT—NOTHING ELSE TO BUY
- INSTALLS IN MINUTES WITH ONLY A SCREWDRIVER
- INTERNAL KITS AVAILABLE FOR*:
  TELEVIDEO
  TANDY 1000
  SANYO
  COMPAQ
  LEADING EDGE
  PANASONIC
  CORONA
  IBM PC
  NOVA
  AT&T, NCR & ITT

10 MEG	20 MEG	32 MEG
$895.00	$1195.00	$1395.00

HARD DRIVE & CONTROLLER ARE 100% IBM COMPATIBLE. WE USE ONLY QUALITY DRIVES & WESTERN DIGITAL CONTROLLERS FOR THE HIGHEST QUALITY. WE BACK THE FULL SYSTEM WITH A 1 YEAR WARRANTY. THE SYSTEM COMES COMPLETE WITH ALL CABLES & MOUNTING HARDWARE. REQUIRES NO ADDED MEMORY BOARD TO OPERATE. 20 & 32 MEGABYTE INTERNAL HARD DISKS ALSO AVAILABLE. COMPLETE INFO AVAILABLE UPON REQUEST.

RTX TECHNOLOGIES, INC.
P.O. BOX 50398
NASHVILLE, TENN. 37209
(615) 373-1251

*SAME PRICE AS TANDY
*ALL MS™-DOS COMPUTERS

"ODE TO PCM"

Editor:

I sat and sobbed, I was dejected
In a world of bytes, I felt rejected
No pulp-mate for my best friend "Candy"
(My name for my sweet 2K from Tandy).

Through endless magazines did scour
Searching yea, on endless hour
Countless pages, filled with hype
Of other computers, not my type.

And then one day, my eyes beheld
My heart within my bossom swelled
I jumped, and shouted "Praise the Lord!"
My "Candy" hadn't been ignored!

I found you, "PCM," at last
And since then, "Candy's" been a blast!
Like "Maidenform," you gave support
To "Candy's" charms. A glad retort!

With "Skedit," "Gallery" and flags array
Wayne Sanders "Webs," blow you away!
"Candy" is happy and so am I
"PCM" delivers more than pie in the sky!

Lou Hefner
San Pedro, CA

Editors Note: Negotiations are underway for a nationally-known music group to record "Ode to PCM." A video is also in the works. Kidding!

2000 MYSTERIES

Editor:

Although the Tandy 2000 is an excellent machine, there are a number of errors in the manuals and many subjects on which it is difficult, if not impossible, to obtain information. Perhaps PCM can act as a clearinghouse for such problems. Some of the following was obtained directly from Tandy after being unable to get answers at local Computer Centers or the new Area Training and Service Offices and may be helpful to other 2000 owners:

1. The codes produced by the 12 function keys, and DELETE, INSERT and HOME keys when used in combination with SHIFT, CTRL and ALT are regular single character codes rather than the extended codes shown in the manuals. In order to use some programs written for the Tandy 1000 or IBM PC, you must trap for these combinations and add the zero character to create the extended code. Tandy states that this is a Microsoft error and should be corrected in the "next version."

2. When using SOUND, the length of the tone cannot be set beyond approximately ten. Therefore, beware of timing loops requiring longer tones. This is to be fixed in a "future version," timing unknown!

3. Cursor start and stop parameters in LOCATE don't work from BASIC, but with assembly language, you can produce either a full block cursor or an underline cursor, but nothing in between.

4. The function key for SCREEN has a third zero which should be utilized for "vpage" or "apage" options. These commands will not be implemented on the 2000 because "the hardware is not able to perform these functions."

5. In assembly language programs, Software Interrupt 11 Hex checks the equipment connected to the computer. Bit 13 for the printer apparently checks for installation of a printer controller board. This hardware is built into the Tandy 2000 (purchased separately with IBM PC) and therefore, any program which checks this bit will always be set.

6. Although you use COM1 to set up the RS-232 buffer from BASIC, you must use AUX to access this port from MS-DOS. For example, the following batch file dials a number:

```
MODE COM 300 (sets RS-232 port at 300
Baud, no parity, 8 bits, 1 stop)
ECHO ATDT 111-222> AUX (sends Hayes
dial command and number to modem)
```

7. The memory map on Page 345 of the BASIC manual, Appendix D, is incorrect for DOS version 2.11.01. With no loadable device drivers or other system parameters in a CONFIG.SYS file, MS-DOS is at 05E0:0000 and the user program area begins at 0B16:0000. The video buffer is the last 5K of memory and the BIOS buffer area takes up the immediately preceding 7K of memory.

Undoubtedly, other readers have found additional problem areas. Thank you for keeping PCM as a source of Tandy 2000 help. It's great to see articles by familiar, knowledgeable writers such as John Harrell, Robert Covington and Bill Barden in PCM.

Charles R. Perelman
Los Angeles, CA

Editors Note: Our readers thank you and we thank you. Keep the informative bits coming!

REPORTS+ COMPATIBILITY

Editor:

We are very pleased to have received our first issue of PCM. We've been Tandy computer fans since 1978 and own a TRS-80 Model I, Model 100 and more recently a Tandy 1200HD.

We bought the 1200 in late December 1984 hoping to gain IBM compatibility. This is fine for 95% of the IBM software we use. Unfortunately, the other 5% is the killer!

Does anyone in this country know how to modify the BASIC programs generated by IBM's *Personal Decision Series Reports+* module? Apparently, *Reports+* sets a memory pointer to IBM BASIC ROM or it does not allocate enough room in which GW-BASIC can reside. I either get an "Error loading GW-BASIC" when GW-BASIC resides in the root directory, or an "insufficient memory" error when it is in the \LIB1 subdirectory.

Needless to say, the *Reports+* module is in the main software product I use at work and would love to have it working on the 1200. *Reports+* works fine, but its generated BASIC programs won't.

If someone could help, it would save us $5,000 for an IBM XT!

Thank you so much for any and all information you may be able to supply!

Bernadette M. Kennelly
Pittsburgh, PA

Editor's Note: You recieved the "Error loading GW-BASIC" message because both *of the GW-BASIC files (*BASIC.COM *and* BASIC.EXE*) need to be in the current directory when you're running a BASIC program. Otherwise, the search path set up with MS-DOS's* PATH *command will find the first file but will be unable to find the second part.*

Once you are successfully in BASIC, it is possible that the program you're using will require more memory on the 1200 than on an IBM. This is because IBM BASIC is partially stored in ROM while the entire BASIC programming language must be loaded into RAM on a Tandy 1200 leaving less room for your application program. Adding memory should solve this problem.

It could be, however, that Reports+ *is acutally trying to use direct calls to IBM's BASIC ROM — a poor practice since a ROM change can make the program obsolete. In this case, there's nothing you can do short of finding new software (or new hardware).*

TMPC

A Plan for Efficiency

For the TRS-80 Model 100

More than a Calendar

TMPC helps you organize your day. It is more than a calendar program; it is a toolbox to help you keep your most urgent tasks at top priority.

Efficiency is no Game

When you use TMPC, you enter the system of screens and corridors shown above and access features by moving through the rooms with the arrow keys. The screens may resemble an adventure game, but TMPC is a serious tool. The user interface will become second nature to you, like walking through the rooms in your home.

Functions for Efficiency

Each screen has its own function. For example, one screen includes a perpetual calendar, and another shows a "warehouse" of tasks to be done. In other screens you can set entries that repeat weekly, monthly, quarterly or yearly; request a warning of up to 999 days for any entry; sort your to-do list by stress level and priority; and more.

Organization Philosophy

TMPC has more than just "features;" it has a *philosophy* of time management inspired by Stephanie Winston's best seller, *The Organized Executive*. You won't learn all there is to know about TMPC in five minutes, but after using it, you'll feel more *organized*, not just more computerized.

Spreading the Word

One user told us that he felt *guilty* to get so much functionality at so low a price. Find out for yourself. Order TMPC today.

TMPC by Ac′rō-ă′tix
$49.95 + $3.00 Shipping

For more information, including how to get a *Free Sample*, call or Write:

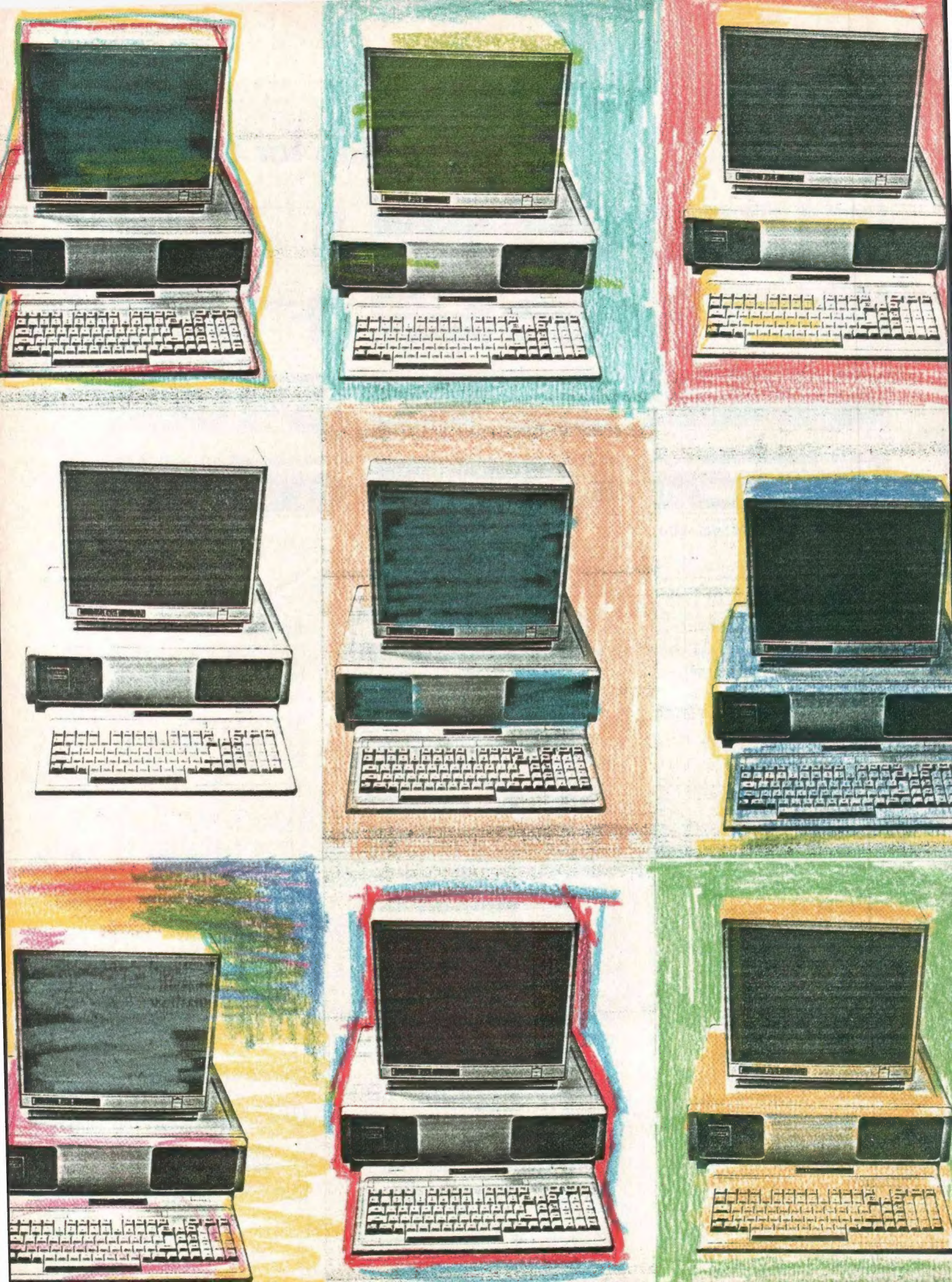
Acroatix Incorporated
P.O. Box 273
Wilmington, MA 01887
(617) 658-5550

Phone orders only call

(800) 225-5800

VISA MasterCard

COD or Courier shipment available at extra charge.



Turn your Tandy's Screen into an Artist's Canvas with this BASIC Program

Sketch

By Kevin Chester

Sketch is a BASIC program that lets the user draw pictures on an editing screen and offers tools to the user to make this simpler. It also offers two ways to save the picture. Drawing pictures in BASIC is tedious, especially if you make a mistake or want to get something "just right." You might have to go through the program, edit and rerun it dozens of times for just one part. The object of this program is to make all this a little easier.

The majority of the functions of *Sketch* are handled through the function keys, while some are handled with letter keys. You can use a keyboard-controlled cursor (small blinking dot moved with the arrow keys) or a mouse-controlled cursor. There is a macro function that remembers a list of commands and repeats them when needed.

The program provides two ways to save the picture. The first is called a "subprogram." This actually duplicates equivalent commands and saves them into a BASIC file (ASCII format) which can later be rerun or merged into a program to duplicate the picture. The other is a memory image save. It stores an image of screen memory to disk so the image can later be recalled in another program.

Start-Up

To start the program, type BASIC and press ENTER at the DOS prompt. When BASIC comes up, type RUN "SKETCH" and press ENTER. There will be a brief message. Press ENTER again, then answer the Model 1000/2000 prompt. Now the program will ask for a filename. This is the "subprogram" initialization. If you answer NONE, *Sketch* will not save anything to disk. If you enter a valid program name, anything you do will be stored to disk under that name as a BASIC program. Simply pressing ENTER will save the program under the default of SAVE.TXT.

Next, the program will ask for palette values to use in the editing process. You should read about palettes in the BASIC manual for more detailed explanations, but here are the fundamentals. There are sixteen colors available, but only eight can be displayed at any one time. When executing a graphics function, BASIC uses palettes to specify colors (there are eight palettes, zero through seven). The user *associates* eight of the sixteen colors to the palettes, one color per palette. When drawing, you specify the palette you wish to use, and the associated color will appear on the screen. Note that you can re-associate the palettes and colors later in the program; if you do, all things previously drawn with the palette in question will also change to the new color. Note also that the background color is usually palette zero. When the prompt appears, enter the number of the color you wish to have associated with each palette.

Now the screen will clear, leaving you with a small blinking cursor in the middle and an arrow for the mouse.

Keyboard Cursor Control

The first commands deal with cursor movement. There are four things the cursor must know: the distance to move, the direction, whether or not to draw a following line and the color or palette currently selected. To choose a distance, press the SHIFT key and a

---

(Kevin Chester is a 16-year-old senior at Miami Palmetto High School in Florida. He is a self-taught programmer who began learning with BASIC on a TRS-80 Model III four years ago, then expanded to 8086 assembly language programming on his Model 2000.)

number (not on the numeric keypad) then press an arrow to choose a direction. The cursor will now move the number of spaces chosen in the direction chosen. For example, press SHIFT-0. This selects ten spaces per move. Now press the up arrow. The keyboard cursor will move ten pixels (or spaces) up.

The l key toggles the line function. To turn the line function on, press the l key. To turn it off, press the l key again. This tells the program that you want a line to follow the cursor in the current color. Press l and move the cursor. To choose the current color, press a number key (from zero through seven on the 2000 and from zero to nine on the 1000). This tells the program that any drawings done are to be drawn in the color associated with the palette number you pressed. Press 4 and draw a line with the cursor. It will appear in the color you specified in the start-up screen.

It is important to make sure that the CAPS key is off. For all applicable commands, the shifted letter will select the command for the mouse and the lowercase letter will select the command for the keyboard. Press L now and move the mouse.

Function Keys

- F1 will provide a list of the commands that all of the other function keys will do when pressed. Press F1. At the top left F1 DISPLAY FUNCTIONS will appear. Press the plus key (+) and the minus key (-) to go up and down the list. Press ESC to return to the graphics screen. Anything drawn in the upper left corner of the display will always be returned after a command has erased it.
- Press F2. This will provide a list of all the colors available. Once again, plus (+) and minus (-) go up and down the list while ESC returns to the graphics screen. If, while in Function 2, a number from zero to seven is pressed, that palette will be changed to the value now showing at the top of the screen. For example, press F2, and then press the plus key (+). Color 1 (blue) will be shown. Press 0 and Palette 0 will be changed to blue. Press the minus key (-) to go back down to zero (black) and press 0 again. The background color will return to black.
- F3 is similiar to F2 in that it provides a way to change the palette values. When F3 is pressed, the upper left is cleared, ready for entry. The format is like this: x1,y1,x2,y2,x3,y3. . . where all x's are palettes and y's are colors. This is the same as the BASIC palette command. For example, 3,6,2,5 will change Palette 3 to Color 6 (brown) and Palette 2 to Color 5 (magenta). Anything drawn with Palettes 5 and 6 have now had their colors re-associated to magenta and brown respectively. Pressing ENTER when finished entering values or pressing ESC at any time will return you to the graphics screen.
- F4 will draw a circle with its center at the keyboard cursor. Enter the radius when prompted. The other prompts control more complex aspects of a the circle. If you want a simple circle, just press ENTER at the prompts you don't understand. Otherwise read the BASIC manual for a more complete explanation. For example, move the cursor somewhere, press F4 and then type 100 at the first prompt. For the other prompts, just press ENTER. A circle with center at the cursor and a radius of 100 will be drawn.

Note: the values that you enter will be remembered. If you do not enter anything for a certain prompt the specification for that prompt will be taken from the last circle drawn. When the program is started, all values are set to default values for a perfect circle. To reset a certain prompt to its default value after changing it, enter 99.99 at the prompt you wish to reset.

- F5 draws a line from the keyboard cursor position at which F5 was first pressed to the keyboard cursor position at which F5 was pressed a second time. Press F5, move the cursor to a new place and press F5 again. A line will be drawn connecting the two points. To change the color of lines drawn, select a new current palette by pressing a number from zero to seven. Lines will now be drawn in the color associated with that palette.
- F6 works exactly the same as F5 except it draws a box with the two opposite corners located at the points selected.
- F7 is the paint function. It will fill any shape that has a complete border with a color. Create a shape (with F4 or F6, for example), place the cursor within the shape and press F7. The first prompt will ask which palette to use to fill in the shape. The second prompt will ask for the palette of the border of the shape. For example, create a circle with palette Number 3. Press F7 and answer 7 to the first prompt and 3 to the second prompt. This tells the function to start filling with palette seven and stop when a border with palette three is found. Note that this function can be aborted by entering 99.99 at either prompt.
- F8 starts the program over and closes the subprogram. More on this later.
- F9 stores a memory image of the screen onto disk which can later be recalled with F10. Press F9 and the computer will ask for a filename. Do not add an extension, just a filename of up to eight characters, press ENTER and wait for it to save.
- F10 will recall a memory image saved with F9. Answer the prompt with the filename under which you saved your picture (up to eight characters and no extension) and wait while it comes up.
- F11 clears the screen and sets the cursor back to the center.
- F12 provides a simple status for the keyboard cursor. It reports on the coordinates of the cursor and whether the line toggle is on or off. Press ESC to exit from here.

Keyboard

Once again, it is important that the CAPS key is off. Many of the keys have two functions, one for lowercase and one for capitals.

- l and L have already been mentioned as toggles for the line function on the keyboard and the mouse respectively.
- p clears the top left and provides a list of the palettes and the color values associated with them.
- s will set a point in the current palette at the current keyboard cursor location.
- r will reset the point at the current keyboard location. Note that resetting is the same as setting in palette 0.
- c will move the keyboard cursor to the position of the mouse cursor.
- C will move the position of the mouse cursor to the keyboard cursor.
- The slash key (/) will clear the upper left and tell what palette value is at the location of the keyboard cursor.
- The question mark key (?) will tell what palette value is at the location of the mouse cursor. As you move the mouse, the display will be updated to show the palette value at each location over which the mouse travels. As an example, press ? and move the mouse around (over some colors). Press ESC to exit these two functions.
- The slash (/) and question mark (?) keys are most useful when you want to fill in a shape already on the screen but can't remember what palette it was drawn with. This is *vital information* for the paint function because it will cover everything within the described border (the palette value you give it).

HARD DRIVE PRICE BREAKTHROUGH

STARTING AT $299.95!

5 to 60 megabytes complete easy to install internal and external systems
— will boot just like an XT — fully warranteed —
same day shipping from stock — fast service — free technical assistance

$499.95 to $999.95
STREAMING OR FLOPPY TAPE BACK UP SYSTEMS
complete internal and external systems

$1,995.00 — IBM XT WORKALIKE SPECIAL
20 megabytes — one floppy — 640K — clock calendar — serial port — parallel port
monochrome card — monochrome monitor with tilt and swivel base
Keytronics 5151 style keyboard — one year warranty — shipped from stock

MEGADISK™ HARD DISK DRIVE SYSTEMS **TOLL FREE ORDERING 1-800-343-8841**

For the { **IBM/PC**, [illegible], **TRS/80 Models I/III/IV/4P**, Compaq, PC Workalikes, **Color Computers**, Heath/Zenith, Max/80
Complete with Hardware, Cables, Software and Quikfit Installation

5 Megabytes Internal Mount	starting at	$ 279.95
10 Megabytes Internal Mount	starting at	439.95
20 Megabytes Internal Mount	starting at	579.95
5 Megabytes External System	starting at	499.95
10 Megabytes External System	starting at	639.95
20 Megabytes External System	starting at	799.95
Tape Backup System — Interna; Or External	starting at	449.95

Wow! New Low Prices

DOS Systems Available: { IBM/Heath — DOS, 1.0, 2.0, 2.1, 3.0, or later
TRS/80-LDOS, TRSDOS 6.x, Newdos/80, Dosplus, CP/M, COCO DOS, Max/80 LDOS, OS9

FULLY WARRANTIED — PARTS AND LABOR — 24 HOUR SERVICE — CALL TOLL FREE — 1-800-343-8841

IBM ENHANCEMENT CARDS

Multifunction with 384K Serial port, parallel port Clock calendar, game port	$259.95	Color graphics	$139.95
Serial port — two port	69.95	Monochrome/RGB Color	229.95
Monochrome only	119.95	Disk controller with cable — up to 4 drives	79.95
Monochrome graphics w/parallel port	159.95	384K memory	129.95
		Hard drive controller	159.95
		Keytronics 5151 style keyboard	129.95
		Network card	249.95

DEALER INQUIRIES INVITED
CALL 1-617-872-9090

PRICES CHANGE EVERY DAY
CALL 1-800-343-8841

PC © 1985 **SOFTWARE SUPPORT, INC.**

1 Edgell Road, Framingham, MA 01701 (617) 872-9090 Telex-383425

Hours: Mon. thru Fri. 9:30 am to 5:30 pm (E.S.T.) Sat. 10 am to 3:30 pm

SERVICE POLICY — Our Professional Technical Staff Is Available To Assist You Monday Through Saturday.
WARRANTIES — Up To One Full Year Parts And Labor. Floppy Disk Drive Power Supplies — Five (5) Years.
SERVICE — 24 Hour Turn-A-Round On All In-Stock Parts. Dealer Inquiries Invited. Call 617-872-9090

Please Call For Shipping, Handling And Insurance.

Toll Free 1-800-343-8841

Please Call For Our Latest Price Saving Specials.

Not Responsible for Typographical Errors. Prices and Specifications May Change Without Notice.

• j will ask for an X-coordinate and a Y-coordinate. The keyboard cursor will be moved to this location.
• w will give you a cursor which can be moved around with the arrow keys. Typing letters on the keyboard makes them appear on the screen at the position of the cursor. Press ESC to leave this.
• x, X and q are all part of the macro function which will be explained later.
• m creates a mark in the BASIC text file.

There are two more keys that are not on the keyboard, but are vital to the mouse. The left key on the mouse operates the same as the F5 key, but it works on the mouse. Press the left key on the mouse and release it. Now move the mouse across the screen, and press it again. A line will appear connecting the two points.

The right key on the mouse operates to negate the effect of the L toggle. If the L toggle is on, then whenever the right key is held down, the toggle will think it is off. Whenever the L toggle is off and the right key is held down, the toggle will think it is on. Try turning the toggle on with L. Now move the mouse around, drawing a line. Press the right key and continue moving the mouse. The line will stop, but when you let the right key up, the line will start again.

The Macro

Sketch provides a macro function; that is, a way of storing a group of commands under one key. It is very useful in reproducing a series of commands for a single object: for example, trees on a map. To start storing commands in the macro, press x. Any commands that put graphics on the screen will be stored. To end storing commands, press X.

Move the keyboard cursor to another spot on the screen and press q. This key is the command to operate the macro. Everything that was stored under the macro is now reproduced at the new cursor location. If the mouse is used to draw, the locations of the lines drawn are remembered relative to the keyboard cursor.

The macro utility will store up to 1,000 commands from the keyboard or mouse. When this is reached, the macro will automatically stop itself and flash a message to that effect.

For example: first press x, then turn on the line function for the keyboard cursor. Now draw a square and fill it in with the paint command. Draw a circle around that, then set the cursor distance to one pixel per move and doodle for a little while. Then press X and move the cursor to a new spot on the screen. Press q and watch the object appear again. If you are drawing maps, this would be great for trees or symbols.

The BASIC Text File

The second way of saving the drawn picture is by writing it directly into a BASIC file which can later be run or merged with other files. On startup, when *Sketch* asks for a filename for program store, this is the function that is operating. Entering none or NONE will tell *Sketch* not to save the picture as a program. Just pressing ENTER will tell *Sketch* to save the picture as a file called SAVE.TXT.

After drawing the picture, press F8 to exit and close the saving file. Then you can run the file you saved (just type run"*filename*" and ENTER). After loading the saved file it is a good idea to save it back to disk, since *Sketch* saves it in ASCII format instead of binary format.

The m key, or mark function, is used here to remember spots in the subprogram. Say you draw a line in the wrong place. It's no problem to just erase it, but that will show up in the subprogram, too. Pressing m will store a "REM * mark number #" where the number sign (#) is a number which increments for each new mark. Now, go back and find the mark and delete the errant lines.

Typing The Program

Sketch was designed to run on the Model 2000 high-resolution eight color (screen Mode 4) or the Model 1000 medium-resolution sixteen-color (screen Mode 5). It is also designed to run with the mouse attached. There are two possible modifications you might want to make to the program. If you do not have the mouse software you must delete all references to the mouse. Also, if you want to make the program run on the Model 1000 medium-resolution four-color (screen Mode 4) to save time and memory when drawing and saving to disk, then all SCREEN 5 references must be changed to SCREEN 4.

Here is a list of mouse references and corrections:

Line 120 — delete (or put behind a REM) CALL to 3820.
Line 480 — delete (or put behind a REM) CALL to 4240.
Line 1460 — delete mouse CALL.
Line 1500 — delete mouse CALL.
Line 2220 — delete mouse CALL.
Line 2260 — delete mouse CALL.
Line 2600 — insert a RETURN statement or delete the CALL in Line 720 since this function requires a mouse attached.
Line 3900 — insert a RETURN statement or delete the CALL in Line 660.
Line 3920 — insert a RETURN statement or delete the CALL in Line 680.
Line 4160 — delete (or put behind a REM) mouse reference.
Line 4180 — delete (or put behind a REM) mouse reference.

That should take care of not having a mouse. Of course, this program can't be fully appreciated without the fluid motions available with a mouse.

If you only need four colors on the Model 1000 instead of the 10 that *Sketch* provides, it might be a good idea to change all of the SCREEN 5 references to SCREEN 4, since working in SCREEN 5 is slower and takes up about four times as much disk space when saving an image.

Here is a list of things to change:

Line 80 — change CLEAR ,,4000,32768 to CLEAR ,,4000.
Line 80 — change PVAL=9 to PVAL=3.
Line 110 — change SCREEN 5 to SCREEN 4.
Line 530 — delete (or protect with a REM) the entire line.
Line 4380 — under FIE1$ change SCREEN 5 to SCREEN 4 and delete the CLEAR statement within the string.

That will make the Model 1000 run in screen Mode 4.

Notes

• *Sketch* requires MS-DOS 2.11.xx or later and BASIC 1.03 or later.
• The CONFIG.SYS file must be initialized to DEVICE=MOUSE.SYS (the driver that comes with the mouse package). If it isn't, the program must have the above mouse references changed.
• The fastest way to move the cursor around is to move the mouse and press c.
• If you are creating a subprogram and doing some fine or close work with the mouse, it would probably be best to use the keyboard cursor with the line function on and the spacing set to one or two, since the line function on the mouse operates continually. If the

mouse moves very slowly, or stops, the subprogram will have a lot of LINE statements in it that draw only dots.

- It is not possible to continue re-edit a picture saved as a subprogram at a later date. What you can do is, right before you press F8 to close the file, save a memory image of the picture. That way, you will have a subprogram getting you to that point, and a memory image of the result on file. Load *Sketch* and and start a new subprogram. Recall the memory image and continue working on it. After closing the subprogram, you can merge the old with the new for a complete picture.
- If you want to display a memory image of your picture in a program of your own, include these program statements:

```
(for the Model 2000):
DEF SEG=&HE000
BLOAD A$+".PL0",0:OUT 416,50
BLOAD A$+".PL1",0:OUT 416,52
BLOAD A$+".PL2",0:OUT 416,49
(for the Model 1000):
DEF SEG=&HB800
BLOAD A$+".SAV",0
```

The variable A$ is the name of the saved file.

- On the Model 1000, make sure you know which screen mode the picture was created under and that you're calling it back under the same one, otherwise you can get some pretty strange results.
- On the Model 1000, don't move too fast, whether it's the mouse, keyboard cursor or function keys. The 1000 just can't keep up with a fast artist.

Variables

A	—	temporary storage.
A$	—	temporary storage.
A(	—	stores current palette values.
ASPECT	—	aspect ratio for circle command.
BOXON	—	flag for F6 indicating start or end of box.
BX,BY	—	origin coordinates for F6 box function.
COLR$(	—	stores color names for F2.
CURC	—	current cursor (or palette) value.
CURCOM	—	current number of statements in the current line of the subprogram.
CURLIN	—	current line within the subprogram.
CURSOR%(	—	storage buffer for top left window.
END1	—	end value for circle function.
FALSE	—	used to determine the logical validity of variables.
FIE1$	—	storage string used to send statements to the subprogram.
FUNCTION$	—	stores the twelve function names for F1.
KY$	—	INKEY$ trap for the main loop.

The listing:

```
5 '*
10 '**************
50 '                        SKETCH.BAS
55 '                  written by Kevin Chester
60 '                  for use on M1000 or M2000
65 '                  Copyright pending 1985
68 '                        version 5.0
70 '              setup
75 KEY OFF:GOSUB 4780:IF MDL=1 THEN 90
80 CLEAR ,,4000,32768!:DEFINT B-D,F-R,T-U:MDL=2:PVAL=9:ASPECT=5/6:GOTO 100
90 CLEAR ,,4000:DEFINT B-D,F-R,T-U:MDL=1:PVAL=7:ASPECT=25/28
100 DEFINT B-D,F-R,T-U:DIM COLR$(15),CURSOR%(1960),FUNCTION$(12),TBL$(20),CURS(1
5,1),MACRO$(1000):DEF FNRT$ (ONE)=RIGHT$(STR$(ONE),LEN(STR$(ONE))-1):DEF FNVL (P
2)=VAL(MID$(MACRO$(MACRO),P2,9))
110 GOSUB 4340:GOSUB 4660:IF MDL=1 THEN SCREEN 3 ELSE SCREEN 5
120 FOR I=0 TO PVAL:PALETTE I,A(I):NEXT:GOSUB 3820
135 '            define starting variables
140 X1=320/MDL:Y1=200/MDL:X2=X1:Y2=Y1:XY=0:CURC=3:TRUE=-1:FALSE=0:LINEON=FALSE:L
INE1=FALSE:RADIUS=50:START=0:END1=6.283186:BOXON=FALSE:CURLIN=100:MARK=1:MOUSLIN
=FALSE:MLINON=FALSE
160 ON KEY(1) GOSUB 880:KEY(1)ON
180 ON KEY(2) GOSUB 1020:KEY(2)ON
200 ON KEY(3) GOSUB 1180:KEY(3)ON
220 ON KEY(4) GOSUB 1560:KEY(4)ON
240 ON KEY(5) GOSUB 1720:KEY(5)ON
260 ON KEY(6) GOSUB 1800:KEY(6)ON
280 ON KEY(7) GOSUB 1460:KEY(7)ON
300 ON KEY(8) GOSUB 2100:KEY(8)ON
320 ON KEY(9) GOSUB 2160:KEY(9)ON
340 ON KEY(10) GOSUB 1980:KEY(10)ON
```

LINE1	—	flag for F5 function indicating start or end of line.	MX,MY	—	current coordinates of mouse.	WAS	— used to make the keyboard cursor blink.
LINEON	—	'l' line function flag.	NUM(	—	used in the F3 (change color) function.	X1,Y1	— current location of keyboard cursor.
M1,M2, M3,M4	—	values passed to the mouse.	OMX,OMY	—	starting position of line for mouse line function.	X2,Y2	— old location of keyboard cursor for line function.
MAC	—	on/off flag indicating status of macro.	PVAL	—	number of palettes available for current model.	X3,Y3	— current location on screen for write function.
MACRO	—	current number in sequence.	QWERTY	—	temporary storage for FOR/NEXT loops.	Xc,Yc	— temporary storage coordinates for 'j' jump function.
MACRO$(	—	stores macro.	RADIUS	—	radius value in circle function.	XDIF,YDIF	— used during the macro procedure to force the macro to draw at the new location.
MARK	—	current mark number.	START	—	start value in circle function.		
MDL	—	model value (1 for Tandy 2000, 2 for Tandy 1000)	SVF	—	flag indicating operation of subprogram.		
MLINON	—	'L' mouse line function flag.	TBL	—	used as a pointer in F3 function (change color).	X1,Y1	— used in F5 to store start of line.
MOUSE	—	used in determining location of mouse software support.	TBL$(	—	stores values in F3 function.	XPLUS	— increment value for keyboard cursor when moving.
MOUSLIN	—	flag for left mouse button, indicates start or end of line.	TRUE	—	used to determine logical validity of statement or variable.	XY	— saved value under keyboard cursor (non-destructive cursor).
MSEG	—	used in determining location of mouse software support.					

```
36Ø ON KEY(11) GOSUB 19ØØ:KEY(11)ON
38Ø ON KEY(12) GOSUB 228Ø:KEY(12)ON
4ØØ ON KEY(13) GOSUB 4Ø2Ø:KEY(13)ON
42Ø ON KEY(14) GOSUB 4Ø4Ø:KEY(14)ON
44Ø ON KEY(15) GOSUB 4Ø6Ø:KEY(15)ON
46Ø ON KEY(16) GOSUB 4Ø8Ø:KEY(16)ON
48Ø GOSUB 414Ø:GOSUB 424Ø
49Ø '            ** MAINLOOP STARTS HERE **
5ØØ KY$=INKEY$:IF KY$="" THEN 48Ø
52Ø IF(ASC(KY$)>47 AND ASC(KY$)<56) THEN CURC=VAL (KY$)
53Ø IF MDL=2 AND (ASC(KY$)>55 AND ASC(KY$)<58) THEN CURC=VAL (KY$)
54Ø IF KY$="l" THEN LINEON=NOT LINEON:GOTO 84Ø
56Ø IF KY$="L" THEN MLINON=NOT (MLINON):GOTO 84Ø
58Ø IF KY$="j" THEN GOSUB 246Ø
6ØØ IF KY$="m" THEN GOSUB 236Ø
62Ø IF KY$="s" THEN GOSUB 238Ø
64Ø IF KY$="r" THEN GOSUB 242Ø
66Ø IF KY$="c" THEN GOSUB 39ØØ
68Ø IF KY$="C" THEN GOSUB 392Ø
7ØØ IF KY$="/" THEN GOSUB 254Ø
72Ø IF KY$="?" THEN GOSUB 26ØØ
74Ø IF KY$="p" THEN GOSUB 27ØØ
76Ø IF KY$="x" THEN GOSUB 3Ø2Ø
78Ø IF KY$="X" THEN GOSUB 3Ø4Ø
8ØØ IF KY$="q" THEN GOSUB 318Ø
82Ø IF KY$="w" THEN GOSUB 348Ø
84Ø FOR I=1 TO 1Ø:IF KY$=COL$(I) THEN XPLUS=I:FOR I=Ø TO Ø:NEXT ELSE NEXT
86Ø GOTO 48Ø'            ** MAINLOOP ENDS HERE **
87Ø '            F(1)-Display Functions
88Ø GOSUB 458Ø:GOSUB 454Ø:P=1
9ØØ LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1:PRINT P" "FUNCTION$(P);
92Ø A$=INKEY$:IF A$ ="" THEN 92Ø
94Ø IF A$="="THEN P=P+1:IF P>12 THEN P=12
96Ø IF A$="-"THEN P=P-1:IF P<1 THEN P=1
```

```
98Ø IF ASC(A$)=27 THEN GOSUB 456Ø:GOSUB 46ØØ:RETURN
1ØØØ GOTO 9ØØ
1Ø1Ø '                    F(2)-Display Colors
1Ø2Ø GOSUB 458Ø:GOSUB 454Ø:P=Ø
1Ø4Ø LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1:PRINT P" "COLR$(P);
1Ø6Ø A$=INKEY$:IF A$="" THEN 1Ø6Ø
1Ø8Ø IF A$="=" THEN P=P+1:IF P>15 THEN P=15
11ØØ IF A$="-" THEN P=P-1:IF P<Ø THEN P=Ø
112Ø IF A$>="Ø" AND A$<"8" THEN PALETTE VAL(A$),P:A(VAL(A$))=P
114Ø IF ASC(A$)=27 THEN GOSUB 456Ø:GOSUB 46ØØ:RETURN
116Ø GOTO 1Ø4Ø
117Ø '                    F(3)-Change Color
118Ø GOSUB 458Ø:GOSUB 454Ø:P=Ø
12ØØ A$=INKEY$:IF A$="" THEN 12ØØ
122Ø A=ASC(A$):IF A=13 THEN 128Ø ELSE IF A=27 THEN GOSUB 46ØØ:GOSUB 456Ø:RETURN
124Ø IF (A>47 AND A<49+PVAL) OR A=44 THEN 126Ø ELSE 12ØØ
126Ø IF P=2Ø THEN 12ØØ ELSE TBL$(P)=A$:PRINT TBL$(P);:P=P+1:GOTO 12ØØ
128Ø TBL$(P)=CHR$(13):P2=P:P=Ø:P3=Ø:A$=""
13ØØ TBL=ASC(TBL$(P)):IF TBL=13 THEN 134Ø ELSE IF TBL<48 THEN NUM(P3)=VAL(A$):A$
=="":P=P+1:P3=P3+1 ELSE A$=A$+TBL$(P):P=P+1
132Ø GOTO 13ØØ
134Ø NUM(P3)=VAL(A$):IF P3/2=INT(P3/2)THEN 136Ø ELSE 138Ø
136Ø LOCATE 1,1:PRINT SPACE$(25);:BEEP:LOCATE 1,1:PRINT"ENTRY ERROR";:FOR I=1 TO
 25ØØ:NEXT:P=Ø:LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:GOTO 12ØØ
138Ø FOR I=Ø TO P3
14ØØ IF I/2=INT(I/2) AND NUM(I)>PVAL THEN FOR I=Ø TO Ø:NEXT:GOTO 136Ø
142Ø IF NUM(I)>31 THEN GOTO 136Ø
144Ø NEXT I:FOR I=Ø TO P3 STEP 2:PALETTE NUM(I),NUM(I+1):FIE1$=FIE1$+"PALETTE "+
FNRT$(NUM(I))+","+FNRT$(NUM(I+1)):A(NUM(I))=NUM(I+1):GOSUB 462Ø:NEXT:GOSUB 456Ø:
GOSUB 46ØØ:RETURN
145Ø '                    F(7)-Paint Area
146Ø GOSUB 458Ø:GOSUB 454Ø:GOSUB 416Ø:LINE INPUT"PALETTE #";A$:GOSUB 456Ø:IF A$=
"99.99" THEN 154Ø ELSE IF (LEN(A$)>1 OR VAL(A$)>PVAL) THEN GOTO 146Ø ELSE A=VAL(
A$)
148Ø GOSUB 454Ø:LINE INPUT"BORDER #";A$:GOSUB 456Ø:IF A$="99.99" THEN 154Ø ELSE
IF (LEN(A$)>1 OR VAL(A$)>PVAL) THEN GOTO 148Ø ELSE AA=VAL(A$)
15ØØ GOSUB 152Ø:GOSUB 46ØØ:GOSUB 418Ø:RETURN
152Ø PAINT(X1,Y1),A,AA:FIE1$=FIE1$+"PAINT("+STR$(X1)+","+STR$(Y1)+"),"+STR$(A)+"
,"+STR$(AA):GOSUB 462Ø:IF MAC THEN MACRO$(MACRO)="7"+STR$(X1)+STR$(Y1)+STR$(A)+S
TR$(AA)+" ":MACRO=MACRO+1:RETURN ELSE RETURN
154Ø GOSUB 418Ø:GOSUB 46ØØ:RETURN
155Ø '                    F(4)-Draw Circle
156Ø GOSUB 416Ø:GOSUB 454Ø
158Ø LINE INPUT"RADIUS ";A$:IF A$=""THEN 16ØØ ELSE IF A$="99.99"THEN 17ØØ ELSE R
ADIUS=VAL(A$)
16ØØ LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1:LINE INPUT"START ";A$:IF A$=""THEN 1
62Ø ELSE IF A$="99.99"THEN 17ØØ ELSE IF (A$="S" OR A$="s")THEN START=5Ø ELSE STA
RT=VAL(A$):IF (START>-6.283187 AND START<6.283187) THEN 162Ø ELSE 16ØØ
162Ø LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1:LINE INPUT"END   ";A$:IF A$=""THEN 1
64Ø ELSE IF A$="99.99"THEN 17ØØ ELSE IF (A$="S" OR A$="s")THEN END1=6.283186 ELS
E END1=VAL(A$):IF (END1>-6.283187 AND END1<6.283187) THEN 164Ø ELSE 162Ø
164Ø LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1:LINE INPUT"ASPECT";A$:IF A$=""THEN 1
66Ø ELSE IF (A$="S" OR A$="s")THEN (IF MDL=1 THEN ASPECT=25/28 ELSE ASPECT=5/6)
ELSE IF A$="99.99"THEN 17ØØ ELSE ASPECT=VAL(A$)
166Ø GOSUB 456Ø:GOSUB 168Ø:GOSUB 418Ø:IF MAC THEN MACRO$(MACRO)="4"+STR$(X1)+STR
$(Y1)+STR$(RADIUS)+STR$(CURC)+STR$(START)+STR$(END1)+STR$(ASPECT)+" ":MACRO=MACR
O+1:RETURN ELSE RETURN
168Ø CIRCLE (X1,Y1),RADIUS,CURC,START,END1,ASPECT:FIE1$=FIE1$+"CIRCLE("+STR$(X1)
```

```
+","+STR$(Y1)+"),"+STR$(RADIUS)+","+STR$(CURC)+","+STR$(START)+","+STR$(END1)+",
"+STR$(ASPECT):GOSUB 462Ø:RETURN
17ØØ GOSUB 418Ø:GOSUB 456Ø:GOSUB 46ØØ:RETURN
171Ø '                    F(5)-Draw Line
172Ø GOSUB 458Ø
174Ø IF NOT(LINE1) THEN LINE1=NOT(LINE1):XL=X1:YL=Y1:GOSUB 46ØØ:RETURN
176Ø GOSUB 178Ø:GOSUB 46ØØ:LINE1=NOT (LINE1):RETURN
178Ø LINE(XL,YL)-(X1,Y1),CURC:FIE1$=FIE1$+"LINE("+FNRT$ (XL)+","+FNRT$ (YL)+")-(
"+FNRT$ (X1)+","+FNRT$ (Y1)+"),"+FNRT$(CURC):GOSUB 462Ø:IF MAC THEN MACRO$(MACRO
)="5"+STR$(XL)+STR$(YL)+STR$(X1)+STR$(Y1)+STR$(CURC)+" ":MACRO=MACRO+1
179Ø '                    F(6)-Draw Box
18ØØ GOSUB 458Ø
182Ø IF NOT(BOXON) THEN BOXON=NOT(BOXON):BX=X1:BY=Y1:GOSUB 46ØØ:RETURN
184Ø GOSUB 186Ø:GOSUB 46ØØ:BOXON=NOT(BOXON):RETURN
186Ø LINE(BX,BY)-(X1,Y1),CURC,B:FIE1$=FIE1$+"LINE("+FNRT$ (BX)+","+FNRT$ (BY)+")
-("+FNRT$ (X1)+","+FNRT$ (Y1)+"),"+FNRT$(CURC)+",B":GOSUB 462Ø:IF MAC THEN MACRO
$(MACRO)="6"+STR$(BX)+STR$(BY)+STR$(X1)+STR$(Y1)+STR$(CURC)+" ":MACRO=MACRO+1
188Ø RETURN
189Ø '                    F(11)-Clear Screen
19ØØ GOSUB 458Ø:GOSUB 454Ø
192Ø LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:INPUT"CLS - SURE";A$
194Ø IF LEFT$(A$,1)="Y" OR LEFT$(A$,1)="y" THEN CLS:GOSUB 46ØØ:RETURN
196Ø GOSUB 46ØØ:GOSUB 456Ø:RETURN
197Ø '                    F(1Ø)-Image From Disk
198Ø GOSUB 458Ø:GOSUB 454Ø
2ØØØ LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:LINE INPUT"ENTER FILENAME: ";A$:IF
LEN(A$)>8 THEN 218Ø
2Ø2Ø IF A$="none" OR A$="NONE" THEN GOSUB 456Ø:GOTO 2Ø8Ø
2Ø4Ø GOSUB 456Ø:IF MDL=2 THEN 2Ø7Ø
2Ø6Ø DEF SEG=&HEØØØ:BLOAD A$+".PLØ",Ø:OUT 416,5Ø:BLOAD A$+".PL1",Ø:OUT 416,52:BL
OAD A$+".PL2",Ø:OUT 416,49:GOTO 2Ø8Ø
2Ø7Ø DEF SEG=&HB8ØØ:BLOAD A$+".SAV",Ø
2Ø8Ø GOSUB 46ØØ:DEF SEG=MSEG:RETURN
2Ø9Ø '                    F(8)-Close File (Restart)
21ØØ GOSUB 458Ø:LOCATE 1,1:PRINT SPACE$(25):LOCATE 1,1
212Ø LINE INPUT"CLOSE SAVE FILE ?";A$:IF A$="y" OR A$="Y" THEN IF NOT (SVF) THEN
 RUN ELSE PRINT# 1,FIE1$:CLOSE:RUN
214Ø GOSUB 456Ø:GOSUB 46ØØ:RETURN
215Ø '                    F(9)-Image To Disk
216Ø GOSUB 458Ø:GOSUB 454Ø
218Ø LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:LINE INPUT"ENTER FILENAME: ";A$:IF
LEN(A$)>8 THEN 218Ø
22ØØ IF A$="none" OR A$="NONE" THEN 226Ø
222Ø GOSUB 456Ø:M1=2:CALL MOUSE(M1,M2,M3,M4):IF MDL=2 THEN 225Ø
224Ø DEF SEG=&HEØØØ:BSAVE A$+".plØ",Ø,32752:OUT 416,5Ø:BSAVE A$+".PL1",Ø,32752:O
UT 416,52:BSAVE A$+".PL2",Ø,32752:OUT 416,49:GOTO 226Ø
225Ø DEF SEG=&HB8ØØ:BSAVE A$+".SAV",Ø,32575
226Ø GOSUB 456Ø:GOSUB 46ØØ:DEF SEG=MSEG:M1=1:CALL MOUSE(M1,M2,M3,M4):RETURN
227Ø '                    F(12)-Status
228Ø GOSUB 454Ø:GOSUB 458Ø:LOCATE 1,1:PRINT SPACE$(25);
23ØØ LOCATE 1,1:PRINT"1:";:IF LINEON THEN PRINT "ON; ";ELSE PRINT "OFF; ";
232Ø PRINT"x-"X1;",y-"Y1
234Ø A$=INKEY$:IF A$="" THEN 234Ø ELSE GOSUB 456Ø:GOSUB 46ØØ:RETURN
235Ø '                    -m- Mark File
236Ø CURCOM=5:GOSUB 462Ø:FIE1$=FIE1$+SPACE$(2Ø)+"REM* mark number"+STR$(MARK):CU
RCOM=5:GOSUB 462Ø:MARK=MARK+1:GOSUB 46ØØ:RETURN
237Ø '                    -s- Set Pixel
238Ø IF MAC THEN MACRO$(MACRO)="8"+STR$(X1)+STR$(Y1)+STR$(CURC):MACRO=MACRO+1
```

```
24ØØ PSET(X1,Y1),CURC:FIE1$=FIE1$+"PSET ("+FNRT$(X1)+","+FNRT$(Y1)+"),"+FNRT$(CU
RC):GOSUB 462Ø:RETURN
241Ø '                    -r- Reset Pixel
242Ø IF MAC THEN MACRO$(MACRO)="9"+STR$(X1)+STR$(Y1):MACRO=MACRO+1
244Ø PRESET (X1,Y1):FIE1$=FIE1$+"PRESET ("+FNRT$(X1)+","+FNRT$(Y1)+")":GOSUB 462
Ø:RETURN
245Ø '                    -j- Jump Cursor To New Location
246Ø GOSUB 454Ø:GOSUB 458Ø
248Ø LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:LINE INPUT"X CO-ORD: ";A$:IF VAL(A$
)<Ø OR VAL(A$)>64Ø/MDL THEN 248Ø ELSE XC=VAL(A$)
25ØØ LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:LINE INPUT"y CO-ORD: ";A$:IF VAL(A$
)<Ø OR VAL(A$)>4ØØ/MDL THEN 25ØØ ELSE YC=VAL(A$)
252Ø X1=XC:Y1=YC:X2=XC:Y2=YC:GOSUB 46ØØ:GOSUB 456Ø:RETURN
253Ø '                    -/- Value At Cursor
254Ø GOSUB 458Ø:GOSUB 454Ø:LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:PRINT"Value a
t cursor:";POINT (X1,Y1);
256Ø A$=INKEY$:IF A$="" THEN 256Ø
258Ø IF ASC(A$)=27 THEN GOSUB 456Ø:GOSUB 46ØØ:RETURN ELSE 256Ø
259Ø '                    -?- Value At Mouse
26ØØ GOSUB 458Ø:GOSUB 454Ø:LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:PRINT"Value a
t mouse:";POINT (MX,MY);
262Ø A$=INKEY$:IF ASC(A$)=27 THEN 268Ø
264Ø OMX=MX:OMY=MY:M1=3:CALL MOUSE(M1,M2,M3,M4):MX=M3:MY=M4
266Ø LOCATE 1,16:PRINT POINT (MX,MY);:GOTO 262Ø
268Ø GOSUB 456Ø:GOSUB 46ØØ:RETURN
269Ø '                    -p- Display Palette Values
27ØØ GOSUB 454Ø:GOSUB 458Ø:P=Ø
272Ø LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:PRINT"P"FNRT$(P)" = "COLR$(A(P))" (
"FNRT$(A(P))")";
274Ø A$=INKEY$:IF A$="" THEN 274Ø
276Ø IF A$="=" THEN P=P+1:IF P>PVAL THEN P=PVAL:GOTO 272Ø ELSE 272Ø
278Ø IF A$="-" THEN P=P-1:IF P<Ø THEN P=Ø:GOTO 272Ø ELSE 272Ø
28ØØ IF ASC(A$)=27 THEN GOSUB 456Ø:GOSUB 46ØØ:RETURN
282Ø GOTO 274Ø
3ØØØ '       **         macro support          **
3Ø1Ø '                    -x- Start Macro
3Ø2Ø IF MAC=TRUE THEN RETURN ELSE MAC=TRUE:MACRO=Ø:RETURN
3Ø3Ø '                     -X- End Macro, Compress
3Ø4Ø GOSUB 416Ø:MAC=FALSE:MACROEND=MACRO-1
3Ø6Ø FOR MACRO=Ø TO MACROEND:MACRO$="":P2=1:FOR P=1 TO LEN(MACRO$(MACRO))
3Ø8Ø IF MID$(MACRO$(MACRO),P,1)><" " THEN 312Ø
31ØØ MACRO2$=MID$(MACRO$(MACRO),P2,P-P2):MACRO2$=MACRO2$+SPACE$(9-LEN(MACRO2$)):
MACRO$=MACRO$+MACRO2$:P2=P+1
```

```
312Ø NEXT P
314Ø MACRO$(MACRO)=MACRO$:NEXT MACRO
316Ø GOSUB 418Ø:RETURN
317Ø '                    -q- Display Macro
318Ø IF MAC THEN GOSUB 3Ø4Ø
32ØØ X2=X1:Y2=Y1
322Ø GOSUB 416Ø:MACRO=Ø:XDIF=FNVL(1Ø)-X1:YDIF=FNVL(19)-Y1:FOR MACRO=Ø TO MACROEN
D:X1=FNVL(1Ø)-XDIF:Y1=FNVL(19)-YDIF
324Ø IF LEFT$(MACRO$(MACRO),9)="5        "THEN GOSUB 336Ø:GOSUB 178Ø:GOTO 346Ø
326Ø IF LEFT$(MACRO$(MACRO),9)="6        "THEN GOSUB 34ØØ:GOSUB 186Ø:GOTO 346Ø
328Ø IF LEFT$(MACRO$(MACRO),9)="8        "THEN GOSUB 338Ø:GOSUB 24ØØ:GOTO 346Ø
33ØØ IF LEFT$(MACRO$(MACRO),9)="9        "THEN GOSUB 338Ø:GOSUB 244Ø:GOTO 346Ø
332Ø IF LEFT$(MACRO$(MACRO),9)="4        "THEN GOSUB 342Ø:GOSUB 168Ø:GOTO 346Ø
334Ø IF LEFT$(MACRO$(MACRO),9)="7        "THEN GOSUB 344Ø:GOSUB 152Ø:GOTO 346Ø
336Ø XL=FNVL(28)-XDIF:YL=FNVL(37)-YDIF:CURC=FNVL(46):RETURN
338Ø CURC=FNVL(28):RETURN
34ØØ BX=FNVL(28)-XDIF:BY=FNVL(37)-YDIF:CURC=FNVL(46):RETURN
342Ø RADIUS=FNVL(28):CURC=FNVL(37):START=FNVL(46):END1=FNVL(55):ASPECT=FNVL(64):
RETURN
344Ø A=FNVL(28):AA=FNVL(37):RETURN
346Ø NEXT:X1=X2:Y1=Y2:GOSUB 418Ø:RETURN
347Ø '                    -w- Write On Screen
348Ø FOR QWERTY=13 TO 16:KEY(QWERTY)OFF:NEXT:GOSUB 458Ø:X3=1:Y3=1
35ØØ LOCATE Y3,X3,1,7
352Ø A$=INKEY$:IF A$="" THEN 352Ø
354Ø A=ASC(A$)
356Ø IF A=27 THEN LOCATE Y3,X3,Ø,7:GOSUB 46ØØ:GOSUB 418Ø:RETURN
```

```
358Ø IF A=28 THEN X3=X3+1:IF MDL=1 THEN 368Ø ELSE 3745
36ØØ IF A=29 THEN X3=X3-1:IF MDL=1 THEN 368Ø ELSE 3745
362Ø IF A=3Ø THEN Y3=Y3-1:IF MDL=1 THEN 368Ø ELSE 3745
364Ø IF A=31 THEN Y3=Y3+1:IF MDL=1 THEN 368Ø ELSE 3745
366Ø FIE1$=FIE1$+"LOCATE"+STR$(Y3)+","+FNRT$(X3)+",1:PRINT CHR$("+FNRT$(A)+")":G
OSUB 462Ø:PRINT CHR$(A);:X3=POS(Ø):Y3=CSRLIN:GOTO 352Ø
368Ø IF X3>8Ø THEN X3=8Ø
37ØØ IF X3<1 THEN X3=1
372Ø IF Y3<1 THEN Y3=1
374Ø IF Y3>24 THEN Y3=24
3742 GOTO 35ØØ
3745 IF X3>4Ø THEN X3=4Ø
3747 IF X3<1 THEN X3=1
3749 IF Y3<1 THEN Y3=1
3751 IF Y3>12 THEN Y3=12
376Ø GOTO 35ØØ
378Ø '                    mouse support routines
38ØØ '                    Initialize mouse
382Ø DEF SEG=Ø:MSEG=256*PEEK(51*4+3)+PEEK(51*4+2):MOUSE=256*PEEK(51*4+1)+PEEK(51
*4)+2:DEF SEG=MSEG
384Ø M1=Ø:CALL MOUSE(M1,M2,M3,M4)
386Ø M1=4:M3=32Ø/MDL:M4=2ØØ/MDL:CALL MOUSE(M1,M2,M3,M4)
388Ø M1=1:CALL MOUSE(M1,M2,M3,M4):RETURN
389Ø '                    -c- Set Kybd Cursor To Mouse
39ØØ X1=MX:Y1=MY:X2=X1:Y2=Y1:RETURN
391Ø '                    -C- Set Mouse To Kybd Cursor
392Ø M1=4:M3=X1:M4=Y1:CALL MOUSE(M1,M2,M3,M4):RETURN
393Ø '                    React To Left Mouse Button
3935 '                    Called From routine @424Ø, Mouse Cursor Support
394Ø GOSUB 416Ø:IF MOUSLIN THEN LINE (MX2,MY2)-(MX,MY),CURC:FIE1$=FIE1$+"LINE ("
+FNRT$(MX2)+","+FNRT$(MY2)+")-("+FNRT$(MX)+","+FNRT$(MY)+"),"+FNRT$(CURC):GOSUB
462Ø ELSE MX2=MX:MY2=MY
396Ø IF MAC THEN MACRO$(MACRO)="5"+STR$(MX2)+STR$(MY2)+STR$(MX)+STR$(MY)+STR$(CU
RC)+" ":MACRO=MACRO+1
398Ø MOUSLIN=NOT (MOUSLIN)
4ØØØ GOSUB 418Ø:FOR I=1 TO 1ØØØ:NEXT:RETURN
4ØØ5 '                    Kybd Cursor Support / Movement
4Ø2Ø GOSUB 416Ø:Y2=Y1:Y1=Y1-XPLUS:IF Y1<Ø THEN Y1=Ø:GOTO 41ØØ ELSE GOTO 41ØØ
4Ø4Ø GOSUB 416Ø:X2=X1:X1=X1-XPLUS:IF X1<Ø THEN X1=Ø:GOTO 41ØØ ELSE GOTO 41ØØ
4Ø6Ø GOSUB 416Ø:X2=X1:X1=X1+XPLUS:IF X1>639 THEN X1=639:GOTO 41ØØ ELSE GOTO 41ØØ
4Ø8Ø GOSUB 416Ø:Y2=Y1:Y1=Y1+XPLUS:IF Y1>399 THEN Y1=399:GOTO 41ØØ ELSE GOTO 41ØØ
41ØØ IF LINEON THEN LINE (X1,Y1)-(X2,Y2),CURC:FIE1$=FIE1$+"LINE("+FNRT$(X1)+","+
FNRT$(Y1)+")-("+FNRT$(X2)+","+FNRT$(Y2)+"),"+FNRT$(CURC):GOSUB 462Ø:IF MAC THEN
MACRO$(MACRO)="5"+STR$(X2)+STR$(Y2)+STR$(X1)+STR$(Y1)+STR$(CURC)+" ":MACRO=MACRO
+1
412Ø X2=X1:Y2=Y1:GOSUB 418Ø:RETURN
413Ø '                    Control Kybd Cursor Display
414Ø GOSUB 42ØØ:XY=POINT(X1,Y1):PSET(X1,Y1),WAS:PSET(X1,Y1),XY:WAS=WAS+1:IF MDL=
1 AND WAS>7 THEN WAS=Ø ELSE IF WAS>3 THEN WAS=Ø
4145 GOSUB 422Ø:RETURN
415Ø '                    Routines To Turn on/off Key Trapping
416Ø FOR QWERTY=13 TO 16:KEY(QWERTY)STOP:NEXT:M1=2:CALL MOUSE(M1,M2,M3,M4):RETUR
N
418Ø FOR QWERTY=13 TO 16:KEY(QWERTY)ON:NEXT:M1=1:CALL MOUSE(M1,M2,M3,M4):RETURN
42ØØ FOR QWERTY=13 TO 16:KEY(QWERTY)STOP:NEXT:RETURN
422Ø FOR QWERTY=13 TO 16:KEY(QWERTY)ON:NEXT:RETURN
423Ø '                    Mouse Movement Support
4235 '                    Called From MainLoop
```

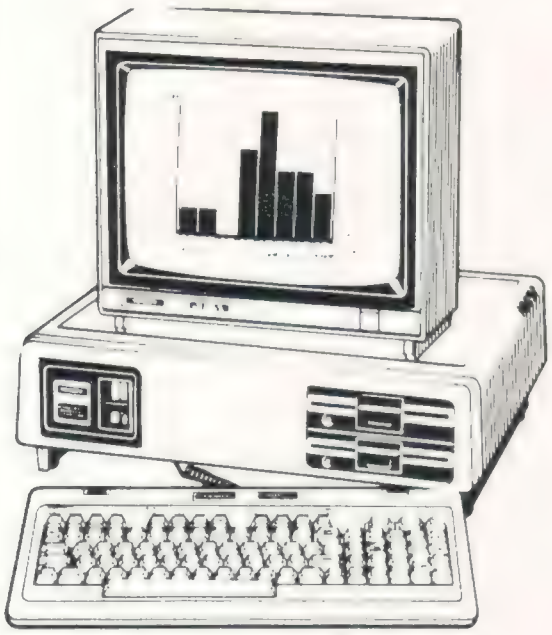
```
4240 OMX=MX:OMY=MY:M1=3:CALL MOUSE(M1,M2,M3,M4):MX=M3:MY=M4:IF M2 AND 1 THEN GOS
UB 3940
4260 IF M2 AND 2 THEN MLINON=NOT (MLINON):GOSUB 4300:MLINON=NOT (MLINON):RETURN
4280 GOSUB 4300:RETURN
4300 IF MLINON AND NOT(MOUSLIN) THEN M1=2:CALL MOUSE(M1,M2,M3,M4):LINE (OMX,OMY)
-(MX,MY),CURC:FIE1$=FIE1$+"LINE ("+FNRT$(OMX)+","+FNRT$(OMY)+")-("+FNRT$(MX)+","
+FNRT$(MY)+"),"+FNRT$(CURC):GOSUB 4620:M1=1:CALL MOUSE(M1,M2,M3,M4)
4320 IF (MAC AND MLINON) THEN MACRO$(MACRO)="5"+STR$(OMX)+STR$(OMY)+STR$(MX)+STR
$(MY)+STR$(CURC)+" ":MACRO=MACRO+1:RETURN ELSE RETURN
4330 '                      Ascii Program Save Initialization
4340 SVF=-1:F$="":CLS:LINE INPUT"enter filename for program store: ";F$:IF F$=""
 THEN F$="SAVE.TXT" ELSE IF F$="NONE" OR F$="none" THEN SVF=0:GOTO 4400
4360 ON ERROR GOTO 4440:KILL F$:OPEN"O",1,F$
4380 CURLIN=20:IF MDL=1 THEN FIE1$="10 cls:screen 3:color 0,1":CURCOM=4:GOSUB 46
20 ELSE FIE1$="10 cls:screen 5:color 0,1:clear ,,,32768":CURCOM=4:GOSUB 4620
4400 CLS:FOR I=0 TO PVAL:LOCATE 1,1:PRINT "value for palette"I;:LINE INPUT A$:A=
VAL(A$):A(I)=A:NEXT
4420 FOR I=0 TO PVAL:A=A(I):FIE1$=FIE1$+"PALETTE "+FNRT$(I)+","+FNRT$(A):GOSUB 4
620:NEXT:ON ERROR GOTO 4460:RETURN
4440 IF ERR><53 THEN ON ERROR GOTO 0 ELSE OPEN"r",1,F$:CLOSE:RESUME
4450 '                      Error Trapping Support
4460 IF ERR=5 THEN I=I-1:RESUME NEXT
4480 IF ERR=53 THEN LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:PRINT CHR$(7);"FILE
NOT FOUND";:FOR I=0 TO 500:NEXT:RESUME NEXT
4500 IF ERR=9 THEN BEEP:GOSUB 4540:LOCATE 1,1:PRINT SPACE$(25);:LOCATE 1,1:PRINT
"MACRO OVER EXTENDED";:FOR QWERTY=0 TO 1000:NEXT:LOCATE 1,1:PRINT"MACRO ENDS AT
THIS POINT";:FOR QWERTY=0 TO 1000:NEXT:GOSUB 4560:GOSUB 3040:RESUME NEXT
4520 STOP
4530 '                      Window Support
4540 GET (0,0)-(199,15),CURSOR%:LOCATE 1,1:RETURN
4560 PUT (0,0),CURSOR%,PSET:RETURN
4580 FOR QWERTY=1 TO 16:KEY(QWERTY)STOP:NEXT:RETURN
4600 FOR QWERTY=1 TO 16:KEY(QWERTY)ON:NEXT:RETURN
4610 '                      Ascii Program Save
4620 IF NOT (SVF) THEN FIE1$="":RETURN ELSE GOSUB 4200:IF CURCOM=4 THEN CURCOM=1
:PRINT# 1,FIE1$:FIE1$=FNRT$(CURLIN):CURLIN=CURLIN+10 ELSE FIE1$=FIE1$+" : "
4640 CURCOM=CURCOM+1:RETURN
4650 '                      Initialization / Data Read
4660 FOR I=1 TO 12:KEY (I) OFF:KEY I,"":READ I$:FUNCTION$(I)=I$:NEXT
4680 FOR I=13 TO 16:KEY (I)OFF:NEXT
4700 FOR I=0 TO 15:READ COLR$(I):NEXT:FOR I=1 TO 10:READ COL$(I):NEXT:RETURN
4720 DATA DISPLAY FUNCTIONS,DISPLAY COLORS,CHANGE COLOR,DRAW CIRCLE,DRAW LINE,DR
AW BOX,PAINT AREA,CLOSE FILE (RESTART),IMAGE TO DISK,IMAGE FROM DISK,CLEAR SCREE
N,STATUS
4740 DATA BLACK,BLUE,GREEN,CYAN,RED,MAGENTA,BROWN,GRAY,BLACK,LT. BLUE,LT. GREEN,
LT. LIGHT CYAN,LT. RED,LT. MAGENTA,YELLOW,WHITE
4760 DATA !,@,#,$,%,^,&,*,(,)
4770 '                      Title Screen and 1000/2000 Request
4780 SCREEN 0:COLOR 15,1:WIDTH 80:CLS:PRINT:CLS:PRINT"Screen oriented sketching
program v 4.50"
4800 LOCATE 5,10:PRINT"By Kevin Chester --- [73075,1502]
4820 LOCATE 23,60:PRINT"press <enter>"
4840 A$=INKEY$:IF A$><CHR$(13) THEN 4840
4860 CLS:PRINT"This program runs on either the Model 1000 or Model 2000.":PRINT:
LINE INPUT"Enter '1' for M1000 or '2' for M2000: ";A$:A=VAL(A$)
4870 IF A<1 OR A>2 THEN 4860
4880 MDL=1:IF A=1 THEN MDL=2    'mdl=1 for M2000; mdl=2 for M1000
4890 RETURN
```

PCM

NOCONA

ELECTRONICS

TRS-80 Model 2000 by Tandy®



TRS-80® Model 100 by Radio Shack

MS-DOS Based

Computer System

256K MOD2000 W/MONO MONITOR	1319.00
256K MOD2000 W/COLOR MONITOR	1629.00
256K MOD 1200 HD 2 DR.	1499.00
128K MOD 1000 W/MONO 1 DR.	839.00
128K MOD 1000 W/COLOR 2 DR.	1199.00
256K MOD 1000 W/HARD DISK	1439.00
24K MOD 100 PORT	469.00
24K MOD 200 PORT	679.00
64K MOD 4D 2 DR	839.00
512K MOD 6000 2DR.	3099.00
80K MOD 12 2 DR.	1199.00
15 MEG HARD DISK	1329.00
35 MEG HARD DISK	2059.00
DMP 105 PRINTER	145.00
DMP 2200 PRINTER	1149.00
DMP 120 PRINTER	249.00
DMP 430 PRINTER	585.00
DMP 2100P PRINTER	1019.00
CGP 220 COLOR PRINTER	499.00
DWP 510 PRINTER	999.00
TRACTOR DWP 210	99.00
TRACTOR DWP II/510	159.00
TRACTOR FEED DMP2100P	99.00
PARALLEL PRINTER SWITCH	96.00
PARALLEL PRINTER CONTROLLER	179.00
PARALLEL PRINTER CABLE	29.00
DISK VIDEO INTERFACE	399.00

100% RS COMPONENTS NO FOREIGN DRIVES OR MEMORY FULL WARRANTY
ALL RS SOFTWARE 20% OFF CATALOG PRICE
CASHIERS CHECK OR MONEY ORDER MUST ACCOMPANY ALL ORDERS

(817) 825-4027

NOCONA ELECTRONICS • BOX 593 • NOCONA, TX 76255

Put some flair into your MS-DOS prompts

Prompts with Style

By Danny Humphress

For the past two months, we have been through an introductory course on using MS-DOS's ANSI.SYS device driver. Today, we'll put some of those ANSI codes to work for us through the use of the MS-DOS PROMPT command.

Cute Little Prompts

By now, the familiar A>, B> and C> prompts are probably good friends of yours. They tell you that MS-DOS is patiently waiting for your next command. And, being the nice guys that they are, they let you know what the current default drive is.

As utilitarian as the cute little MS-DOS prompts are, they are sometimes lacking in . . . well, charisma. Yes, I hate to say it, but our little prompt friends are quite boring. An occasional "Your Wish is My Command>" prompt or "What Now?" would certainly make those long nights at the computer a little less tedious.

What about when one has to set up a computer to be used by a novice? There's nothing like a cryptic A> prompt staring a new user in the face to perpetuate the stereotype of the big mean computer. Something along the order of "Howdy! What'll it be?" is much more in tune with these computers' Texan roots. The prompt could even say something like, "Type HELP if you need it" as a comfort to the new user.

Well, all this and more is possible thanks to MS-DOS's PROMPT command.

PROMPT allows you to change the MS-DOS prompt to just about anything you like. It can even contain the current drive letter, the directory path, the date and the time. Imagine a bank-marquee-like prompt: "At 10:50:00 the default drive is A."

With the use of ANSI codes, you can even have the prompt change screen colors and other attributes. If you want the screen to always be white on blue, you can have the MS-DOS prompt change the attributes to those colors each time it is displayed. This way, if you go into a program that changes the colors, they will return to your preferred colors when your prompt appears.

Darned Easy

One of the best things about PROMPT is that it is so darned easy to use. Suppose we want to change the prompt to "Now What?." The command would be:

```
PROMPT Now What?
```

The next prompt you see will be your own creation!

Special Characters

Some characters (especially ANSI codes) cannot be entered directly when using PROMPT. MS-DOS has provided a way around this though, by using special character sequences. Just as ANSI codes are preceded with an ESC character, special PROMPT codes are preceded with a dollar sign ($). Here is a list of those codes:

Code	Prompt
$$	The $ character
$t	The current time
$d	The current date
$p	The current directory
$v	The MS-DOS version number
$n	The current drive
$g	The > character
$l	The < character
$b	The | character
$_	A carriage return and line feed
$s	A leading space
$h	A backspace
$e	The ESC character

As you know, the "normal" MS-DOS prompt is the drive letter followed by a greater-than character (>). If we put this in PROMPT terms, the command would be:

```
PROMPT $n$g
```

The $n causes the prompt to display the current drive and $g displays a greater-than character (>). If you change the default drive, MS-DOS will be smart enough to change the prompt accordingly. If you had used A instead of $n, the prompt would always say A> — not very desirable unless you want to really confuse someone!

A prompt that I personally like to use is:

```
PROMPT $p$g
```

This prompt shows the current directory path (including the drive letter) and a greater-than symbol. If, for example, you were in a directory called WORDSTAR on Drive C, the prompt would be C:\WORDSTAR>. I find this especially useful in that you are never in doubt about which directory you're currently in.

Some people who have just recently learned how to use PROMPT tend to go *super* fancy. They'll use prompts that show the date, time, directory, MS-DOS version and their own little message. That's OK, though, *you're* the one who has to live with it! And besides, it's a good way to learn about PROMPT while having fun at the same time. Try this prompt on your computer to see what I mean:

```
PROMPT It's $t on $d$_You're in
$p$_Ready$g$_
```

Don't confuse the underline characters (_) with the hyphen (-). The command code $_ causes the prompt to advance to a new line.

You may have noticed that $$ prints a single dollar sign ($). If you were to use a single dollar sign in the PROMPT command line, MS-DOS would think that it was part of one of the special codes. MS-DOS knows to print a single dollar sign ($) in the prompt wherever you used $$.

Another special circumstance is when you want the prompt to start over a number of spaces from the left margin of the screen. If you just entered spaces between PROMPT and your text, MS-DOS would ignore them. The way around this is to use $s for each leading space you want. For instance, to indent the prompt ten spaces, use ten $s's. After you've entered a character other than spaces, MS-DOS will start paying attention to the spaces. So actually, you only need to enter one $s and nine spaces — the spaces after the $s will be printed normally.

ANSI Prompts

As mentioned earlier, it is possible to embed ANSI code sequences into MS-DOS prompts. One interesting use for this, if you have a color monitor, is to use the prompt to keep the screen colors changed to ones you like. I, for example, have a magenta prompt showing the directory path and commands are entered and displayed in blue. This is done by changing the color to magenta before displaying the prompt and *then* changing the color to blue. Here's my favorite prompt:

```
PROMPT $e[35m$p$g$e[34m
```

Look complicated? Well, if you take it one code at a time, it's not so bad. The $e[35m sends an ESC character ($e) and the ANSI code for magenta ([35m). The $p and $g display the current path and a greater-than character (>). Finally, $e[34m sends an ESC character followed by the ANSI code for blue.

After entering this prompt, what you type in MS-DOS and what is displayed on the screen will be in blue. The prompt will be in magenta.

Now, my taste in colors runs slightly left of normal. So, I won't be offended if you tone them down to your own tastes. But as I said, *I* am the one who has to live with it!

If you have a color Tandy 2000, you'll not be able to get colored prompts without first changing the screen mode to color. This is done by sending the ANSI code, ESC [3h to the screen once. Use the BASIC program from last month or use a PROMPT $e[3h to get it in the proper mode and then enter your new prompt codes. You would not want to put $e[3h in your normal prompt, since it clears the screen *each time* it's sent to the screen.

With a little creativity, you can create some marvelous and exciting prompts. It kind of makes me feel sad for our true-blue (make that *white*) A>, B> and C> friends. But, what the heck. Computers don't have feelings — do they?

Seriously, PROMPT is the one command in MS-DOS that lets you really express your creativity. Have a ball!

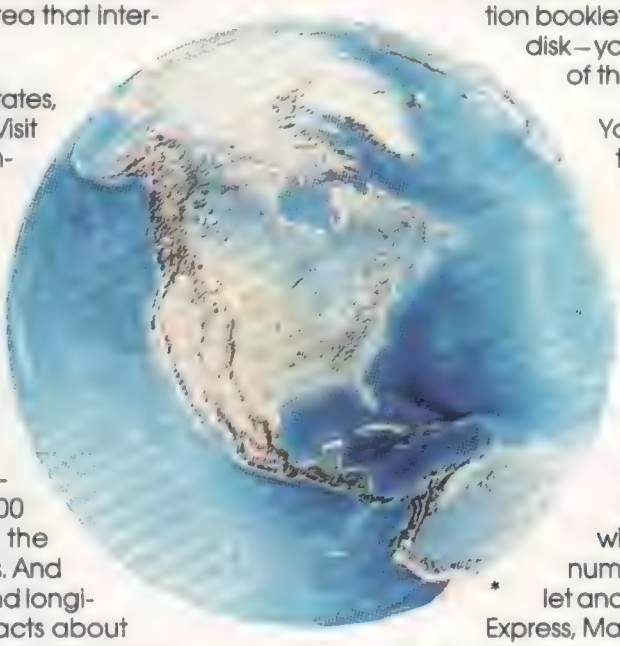
PCM

Give us $5.00 and we'll put the world on your Tandy PC!

Quick, what's the capital of Sri Lanka? What countries do you pass through traveling from Copenhagen to Rome? For learning, explaining, or just plain daydreaming, there's nothing like an atlas. Now, with **CONCEPTS COMPUTERIZED ATLAS**, the world enters the computer age. You can too. In your own home and for only $5. Thanks to our unique Phoenix Plan.

What Is a Computerized Atlas?

You can forget the bulky volume with the endless index. We put a full-color, three-dimensional globe on your screen. Turn it east, west, north, or south. Select an area that interests you and enlarge it.



Visit any state in the United States, any province in Canada. Visit any country in the world! Simply type the name, and the **Atlas** brings it to the center of the screen. A closer look? Magnify it twice, three times, five times! Call up the Fact Sheet to find the population, the size, the capital, the monetary unit and more.

Our **Atlas** can give you information on more than 2,500 cities. Just type the name, the **Atlas** shows you where it is. And tells you the city's latitude and longitude, its population, and facts about the country it's part of. Big cities, historical cities, vacation spots, places in the news.

With our **Atlas**, you can even find the distance between two places in the world, a feature no bulky volume can duplicate. And the **Atlas** can calculate it in land miles, nautical miles, or kilometers. How far is it from London to Moscow in kilometers? Iceland to the Virgin Islands in nautical miles? It's easy with our **Atlas**.

Our **Atlas** was compiled from a number of sources, including a CIA satellite. Over three years of research have gone into our **Atlas** to make sure that it has the latest, most up-to-date information available.

System requirements: Tandy 2000, 1200, or 1000, 128K of memory, a color graphics board, and a color monitor.

Of course, you have to try our **Atlas** to appreciate it. We know that. So we have devised the Phoenix Plan to let you do just that—and for little more than the cost of a blank disk.

CONCEPTS COMPUTERIZED
Atlas

The Phoenix Plan

The Plan is simple. Try our **Atlas** *before* you buy it. For only $5 you get the complete **Atlas** software and with it our easy-to-follow instruction booklet. This is *not* a demonstration disk—you have all the capabilities of the **Atlas** at your fingertips.

You can load the program three times and use it each time for as long as you like. After the third time, the program locks. If you decide not to keep the **Atlas**, simply format the disk, and you have a blank quality disk. The blank disk alone is a $4 value.

But if you want to keep **Atlas** (and we think you will), simply dial the toll-free number in the instruction booklet and charge it to your American Express, MasterCard, or VISA. The operator will take your order and give you the unlocking code over the phone. Our **Atlas** costs only $69.95. (Of course, we'll deduct the $5 you paid for the Phoenix.)

Our patented Phoenix Plan has been successfully used to market our business software by Lanier, Philips and Xerox—to name a few. Now, for the first time we are making the Phoenix Plan available to the PC marketplace. So why don't you try it? Fill out the order form below and send it with your $5 today. You've got nothing to lose and the world to gain. Available at participating Software Concepts dealers (inquiries invited).

YES, put the world on my Tandy PC. I enclose my $5.00. Please send the **Atlas** to:

Name _____

Address _____

City _____

State/Zip _____

☐ **Tandy 2000** ☐ **Tandy 1000/1200**

Mail to Software Concepts, P.O. Box 3323, Wallingford, CT 06494

* Rand McNally & Co., R.L. 84-GP-11

Assembly Language on the Tandy MS-DOS Computers

By William Barden, Jr.
PCM Contributing Editor

"Could you write an 8000-word article that would be a good introduction to assembly language on the Tandy 1000, 1200 and 2000?" the PCM staffer inquired. "Ah . . . well . . . in 8000 words, that's a little difficult. Assembly language is tough! It's not something you can explain in one article," I hedged. "Come on, give it a try," the staffer urged.

The result is this column and next month's column. What I've tried to do in the two is strip assembly language down to the bone, and provide a tutorial that will show you how to write short assembly language code that will interface to BASIC. You won't even need an expensive assembler program ($99.95 from Tandy) to learn some of the techniques here. Since I've already used 121 words, we'd better hurry on . . .

(William Barden, Jr., is a master communicator in a field in which he is one of the few recognized experts —microcomputers. A prolific author of more than 27 books and handbooks on computers and computer programming, Bill also has authored several instructional software projects for Tandy/Radio Shack.)

What Assembly Language Is

The 8088 microprocessor in the Tandy 1000 and 1200 and the 80186 microprocessor in the Tandy 2000 contain a built-in instruction set of simple instructions called *machine language*. Ultimately, every program that runs on these systems boils down to a sequence of machine language instructions, from the BASIC interpreter to *Lotus 1-2-3, dBASE* and *Microsoft Word*. Even though it may appear that instructions are being executed in a higher-level language such as BASIC or PASCAL, every function becomes a low-level machine language instruction, such as "add two eight-bit numbers," "jump to Location 'X' if the result of the last subtract was zero," and "read in a byte from the communications port."

The 8088 and 80186, by the way, contain a nearly identical instruction set, so we'll be using the generic term "8088" to describe both the Tandy 1000, 1200 and Tandy 2000.

A typical short program to add the numbers from one through 100 could indeed be coded in machine language as a series of ones and zeros, as shown in Figure 1. Such coding was actually done in the early days of computers. Ten thousand bleary-eyed programmers later, however, a new technique was developed.

Assembly Language Coding

Assembly language replaced the tedious machine language programming of ones and zeros with a more English-like input. Assembler programs translated mnemonics representing the machine-language instruction set into the corresponding machine language code automatically, as shown in Figure 2. Programmers could now code in *symbolic* code rather than binary ones and zeros or hexadecimal. The assembler program processed the assembly language code in much the same way as a BASIC compiler processes BASIC statements.

The Tandy systems have a special, sophisticated assembler program called a *Macro Assembler* that processes programs written in assembly language. It provides all kinds of bells and whistles, including the capability of a type of "in-line" subroutine called a macro. However, for our purposes, we'll use the assembly capability of the DEBUG program found on all Tandy MS-DOS system disks. DEBUG contains a mini assembler that is equivalent to the first assemblers developed for computers. More importantly, it pro-

vides a way to quickly enter assembly language code, assemble it and debug it. If this is done on the *Macro Assembler*, the process is much more tedious because the assembly language text must be entered via EDLIN or another text editor, loaded and assembled by the *Macro Assembler*, a listing printed and the result of the assembly — an "object module" — loaded into memory by a "link load" process. The convenience of the DEBUG assembler is perfect for the first-time assembly language student.

Loading Debug

The journey of a thousand instructions starts with the first byte. The first step in learning assembly language is to load DEBUG from MS-DOS:

```
A>debug
```

After loading you'll see a minus sign. This is a "prompt" character indicating that DEBUG is ready for the next command.

```
A>debug
-
```

CPU Registers

The first command to try is R, for register. After pressing the R key (lowercase or uppercase, it doesn't matter), you'll see a display like Example 1.

Because we're dealing with three systems here, some of the displays I've indicated may display different address values or memory contents values than yours. As long as the displays look more or less the same, you're in fine shape! The displays here are those that appear on a 128K Tandy 1000.

The R command displays the 8088 *registers*. The registers in the 8088 are really just high-speed memory locations. They are used to hold temporary data, such as the results of adds, subtracts, shifts and input/output operations. All data transferred between the 8088 CPU and user memory generally passes through CPU registers. Registers are not addressed by numbered locations, although they could be. They are also not part of memory, but exist as separate locations in the CPU. The registers are called by letter designations as shown in the display, which lists all CPU registers in *hexadecimal* form.

All registers are 16 bits (two bytes) in width, unlike memory locations which are organized in 8-bit (one byte) widths. Before we go on, I have some bad news for you — you must learn to use *hexadecimal* representation to work with assembly language. Cheer up though — it isn't half bad. You know that sixteen bits involve a combination of sixteen ones and zeros. Hexadecimal is just a shorthand way of representing binary. Every four bits is converted into a single hexadecimal digit as follows:

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Use the BASIC HEX$ command to convert between decimal and hex for practice. PRINT HEX$(10000), for example, prints the Hex equivalent of decimal 10,000 and the &H prefix converts the other way (PRINT &HFF34, for example, prints the decimal equivalent of Hex FF34). You'll also find conversion tables in many reference books.

You'll notice from the display that each register is represented by *four* Hex digits, representing four groups of four bits. All data entered into a register or instruction must be in hexadecimal. To see how this works, try entering data into the AX register and watch it change. To enter data into any register, just type R followed by the register name:

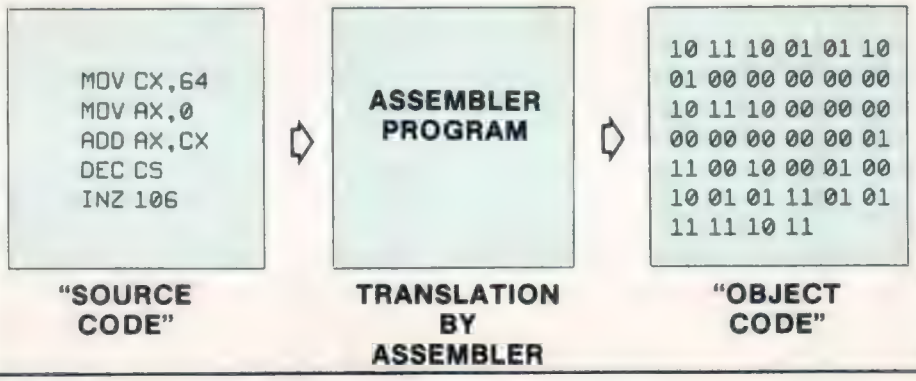
```
-rax
AX 0000
:1 ENTER
-rax
AX 0001
:
```

Here we've displayed the AX register, observed that it contained 0000 (binary 0000000000000000), changed the contents to one and then displayed the register again. The colon indicates that

Figure 1
Machine Language Coding

BINARY CODE	INSTRUCTION
10 11 10 01 01 10 01 00 00 00 00 00	Load CX with 64
10 11 10 00 00 00 00 00	Load AX with 0
00 00 00 01 11 00 10 00	Add CX and AX
01 00 10 01	Subtract from CX
01 11 01 01 11 11 10 11	If CX not 0 go to Add

Figure 2
Symbolic Assembly Language Coding



Example 1

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0930 ES=0930 SS=0930 CS=0930 IP=0100 NV UP DI PL NZ NA PO NC
00930:0100 0000     ADD  [BX+SI],AL                    DS:0000=CD
```

Text processing power that no other program can equal.

PCSG says "Send it back in 30 days for a full refund if you don't agree."

on Snap-in™ Cartridge $149.95

WRITE ROM is the definitive word processing extension for the Model 100. PCSG the first text formatter for the Model 100, now sold by Radio Shack as Scripsit 100. Now 18 months later PSCG introduces WRITE ROM. Those who have experienced it say WRITE ROM literally doubles the power of the Model 100.

First of all, WRITE ROM as its name implies is on a snap-in ROM. You simply take a quarter and open the little compartment on the back of the Model 100 and press it in. It is as easy as an Atari game cartridge. You can use other ROM programs like Lucid whenever you wish.

WRITE ROM lets you do every formatting function you would expect like setting margins, centering, right justifying and having headers and footers. But it does them under function key control.

WRITE ROM remembers your favorite format settings so that you can print a document without any setup, but you can change any formatting or printing parameters instantly with a function key.

WRITE ROM's "pixel mapping" feature shows you an instant picture on the screen of how your printout will look on paper.

In all there are 64 separate features and functions that you can do with WRITE ROM, and some of these features are truly breakthroughs for the Model 100.

First, WRITE ROM lets you do search and replace. Any word or phrase in a document can be searched for and replaced with any other phrase where the search words appear.

Second, WRITE ROM lets you send any text (formated or not) to any other computer over the phone with just a function key. What's more it dials and handles sign on and sign off protocol automatically.

Third, WRITE ROM has a wonderful feature called Library that lets you record favorite phrases, words or commonly used expressions (often called boilerplate).

Any place you wish any Library text to appear you just type a code. WRITE ROM automatically inserts the text just like a Xerox Memory Writer.

Picture what you can do with that kind of capability.

WRITE ROM is blindingly fast. No one can claim faster operation. Because it is on ROM it uses virtually none of your precious RAM. It works with any printer, serial or parallel. You can make a duplicate copy of a document file under a new filename. Rename or delete (kill) any RAM file with function key ease.

This description only scratches the surface of the amazingly powerful piece of software. Dot commands allow control of such things as margins, centering, line spacing and other changes in the middle of a document. Most are Wordstar compatible.

A mail merge feature allows you to send the same document to every name on your mailing list, personalized for each recipient.

WRITE ROM enables you to do underlining, boldface and correspondence mode as well as any other font feature like superscripts that your printer supports in a way that many users say "is worth the price of the program."

To underline you don't have to remember a complicated printer code. You just type CODE U, and to stop underline, CODE U again. The CODE key is to the right of your spacebar. Boldface? CODE B to start and stop. Easy to remember and do. Five different printer features of your choice.

We couldn't list all the features here. For example, not just double space but triple or any other. You can use your TAB key in a document. WRITE ROM allows you to indent. This means you can have paragraphs that have a first line projecting to the left of the rest of the paragraph. Plus many more features.

WRITE ROM has a feature unique to any word processor on any computer. It is called FORM. FORM is an interactive mechanism that lets you create screen prompts so that you or someone else can answer them to fill out forms or questionaires.

With FORM anyplace where you had previously typed a GRAPH T and a prompt in a document, WRITE ROM will stop and you are shown that prompt on the screen. You can type in directly on the screen and when you press F8 you see the next prompt. Goes to a printer or a RAM file.

Think of how you can use FORM. A doctor or nurse could use it for a patient's history with each question appearing on the screen. An insurance salesman could have his entire questionaire. You could construct a series of prompts to answer correspondence typing the answers, even using Library codes. This feature lets you answer letters in rapid fire fashion each with personalized or standard responses.

Before WRITE ROM you had to be a programmer to create a series of prompts. Now its as simple as GRAPH T.

PCSG makes the claim that WRITE ROM is the easiest, fastest and most feature rich formatter for the Model 100. We are happy to offer WRITE ROM because it expands the 100 to a dimension of text processing you cannot equal on even larger computers.

We brashly state that WRITE ROM is the best you can buy. But put that to the test. If you aren't as excited as we are return it for a full refund. Priced at $149.95 on snap-in ROM. MasterCard, VISA, American Express and COD.

1-214-351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd., #207, Dallas, Texas 75229 • 1(214) 351-0564

PCSG provides hotline software support for the Model 100 Call us at 1-214-351-0564

Hex data can be entered to change the register; if you want to leave the register as is, just press ENTER.

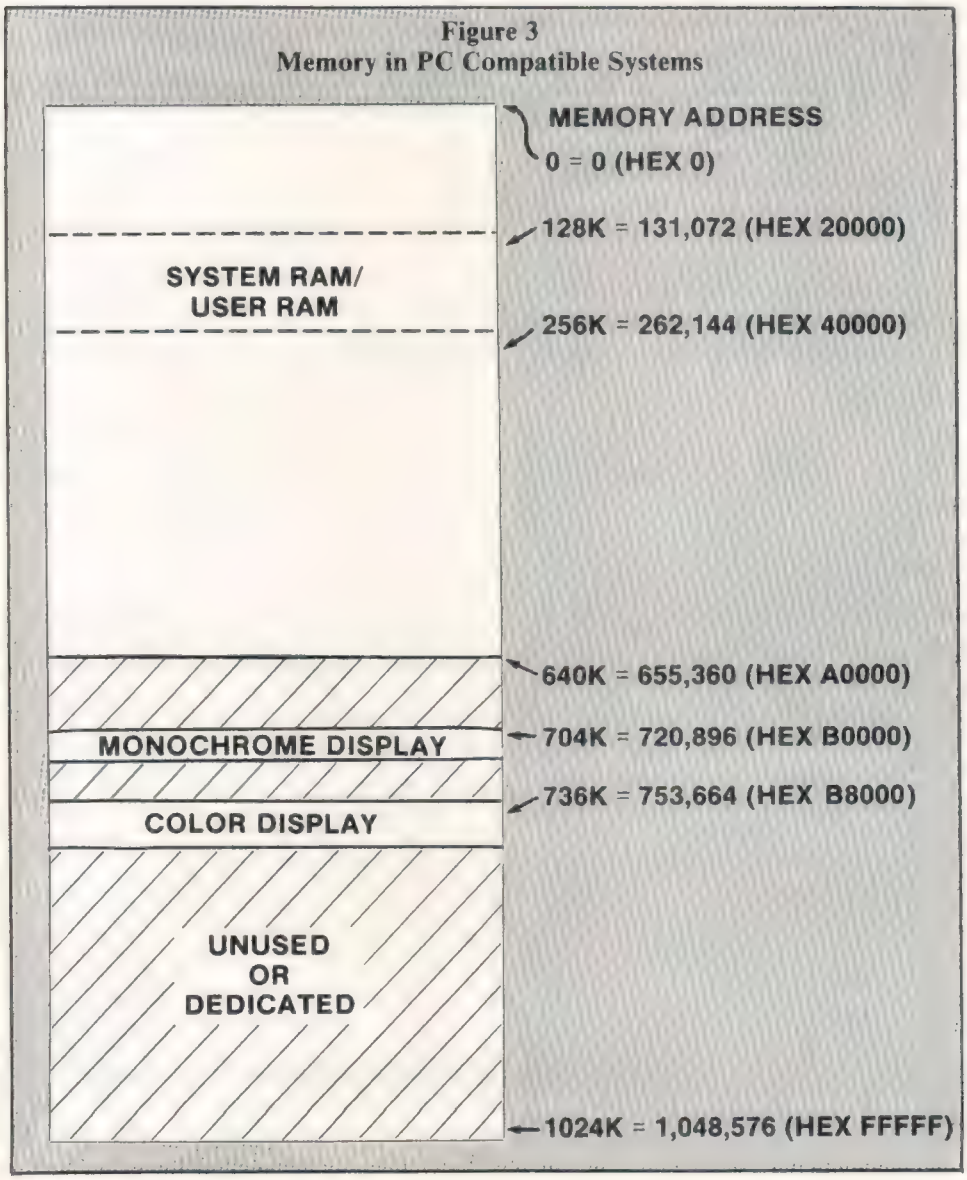
The AX, BX, CX and DX registers are used as "working registers." They hold the instruction-by-instruction results of operations such as adds, subtracts, multiplies, shifts, ands and ors.

The BP, SI and DI registers are "index registers" used in some of the more esoteric addressing modes of the 8088.

The IP register is the instruction pointer register. It points to the next byte of the instruction to be executed. In the display above, IP holds 0100, indicating that the byte of the next instruction to be executed is at location 0100. This corresponds to the display of 0930:0100 opposite the ADD on the third line of the R display. Instructions are arranged in sequence, just like the statements of a BASIC program, and generally executed one after another with occasional jumps to other locations. In case you're wondering about the ADD [BX+SI],AL, by the way, it is an "unassembly" of the data that DEBUG found at location 0930:0100. DEBUG decodes data into the equivalent 8088 instructions. In this case, location 0930:0100 holds a 0000, equivalent to the instruction ADD [BX+SI],AL. Since no instruction has been entered, the 0000 is really just garbage that DEBUG has attempted to decode into something meaningful.

The CS, DS, SS and ES registers are "segment registers." They point to a 64K (65,535) byte block of memory known as a "segment." Memory in the 8088 is divided up into 1,048,576 bytes of memory as shown in Figure 3. In PC compatible systems, only the lower 640K bytes are used for user RAM memory. The remaining memory is used for video display memory, ROM memory and dedicated addresses for input/output devices. The memory addressing space is divided up into 64K byte segments used for the code (code segment), for data (data segment), for the stack (stack segment) and for user defined functions (extra segment). The segment registers point to the 64K byte block. From the display above, you can see that in this case, all segment registers point to the same 64K byte segment — 0930. There's no reason that the same segment can't be used for instructions, for data and for the stack. This is often done, as for example, in interpretive BASIC, which uses a 64K segment.

Figure 3
Memory in PC Compatible Systems



Memory Addresses

The actual physical memory address referenced is made up of the segment address and an "offset." The segment address is the upper 16 bits (four Hex digits) of the start of the physical block. In the case above, for example, 0930 represents the physical address of 09300 in the PC system, the start of the 64K block. The complete address is found by adding the segment address and the IP offset together. In this case, 09300 plus 0100 equals 09400, the actual physical address of the ADD [BX+SI],AL "dummy" instruction.

The form of the address 0930:0100 indicates that the segment is at 09300 and that the offset is 0100. You can simply use the offset alone within the same segment, and this is often done.

The "stack" area of memory is an area of several dozen to several hundred bytes devoted to an assembly language stack. The stack builds downward in memory, as shown in Figure 4. The stack is used to store return addresses for assembly language calls (similar to GOSUBs in BASIC), temporary results for PUSH instructions, and, in more advanced assembly language programs, for interrupt addresses and data. As most assembly language programs will use subroutine CALL instructions, a stack is required. DEBUG sets the SP (stack pointer) register to an offset of FFEE initially, with the stack segment starting at 0930, the same as the other segments. The FFEE indicates that the stack starts *before* the 0930 start of the segment; we'll provide some examples of stack use further on in these columns.

Assembly Language Instructions

As we mentioned before, the 8088 has a built-in set of instructions that perform rudimentary operations. Each instruction has a *mnemonic* associated with it. The mnemonic is a shortened form of the description of the instruction. Instead of entering "add the contents of the BX register to the contents of the AX register and store the result in the AX register," the text ADD AX,BX is used. If we wanted to enter

It's the Holmes Engineering/PCSG "chipmunk"

PORTABLE Disk Drive

for the Model 100/200

$599
Includes a library of six powerful programs

Uses the main menu concept. You see the disk directory instantly, arranged on your M-100 screen like your main menu. Just move the widebar cursor and transfer files with a function key. You can run a file directly from the diskette with the ENTER key. Uses 3½" microfloppy diskettes that have a rigid plastic casing and a metal core. They're tough and nonflexible. You can carry several in a shirt pocket without damage. There's 358K on a diskette. Ten of these in your briefcase and you've got 3½ megabytes.

Drive weighs only three lbs. and it works directly from the 110 outlet and recharges at the same time. It recharges in six hours with thousands of pages transferred between charges. It's compact, with dimensions of 2¼" x 5½" x 7.5"; and fits easily into your briefcase along with your Model 100 or 200.

Machine code programs, BASIC programs, *Lucid* files and documents all are saved and retrieved with no protocol–instantly, ready to run.

In a special association, Holmes Engineering and PCSG have worked together combining the hardware knowledge of Holmes and the software expertise of PCSG. The result is a product that can only be regarded as excellent.

You see the disk directory instantly; works just like the main menu

Here is what is really exciting. The portable disk drive has Random Access. Included as part of the operating system in the drive (ROM) is a very powerful disk BASIC.

This means that you can have BASIC programs that will access the diskette and read and write records directly on the diskette.

Just imagine yourself with this kind of capability.

Database – The portable disk drive stores your mailing list, inventory items, part #s and descriptions or any other data that you need to recall.

358K on a diskette

Invoice (purchase order) – At the touch of a button you can print out your sequentially numbered, professionally done invoices. This is truly professional invoicing capability.

Purchase orders are just as easy.

Sort – This excellent utility allows you rapid sorting of any records you have compiled. You can write the newly sorted list back in the same file on the diskette or to a new file.

Telecom interface – If you are a user that likes to access other computers or databases (for example CompuServe) by telephone then this powerful facility alone is worth the price of the disk drive. You can automatically download and upload information onto the diskette.

Calendar – Everyone who has seen this program has said, "This is the first calendar/diary/scheduler on any computer anywhere that I can use. It is so functional."

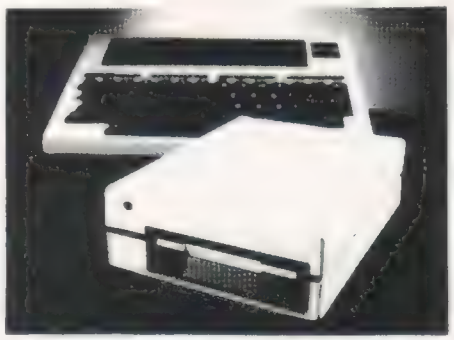
The calendar program is usable for two reasons, first it is designed correctly, and second you have the memory (358K) on the diskette to log and access a tremendous amount of notes over a long period of time.

Personal Finance Manager – This wonderful program truly lets you keep track of your finances.

All your records are kept on the diskette. Bank accounts (checking and savings) and charge accounts such as MasterCard and Visa.

We at PCSG believe we have the ultimate Model 100 system, the Portable Disk Drive plus the *Lucid* spreadsheet on snap-in ROM, *Write ROM* word processing and the new 64K RAM expansion now available from PCSG.

We want you to find out for yourself at no risk. If you aren't totally satisfied within 30 days, simply return the disk drive for a full refund. Priced at $599.95, including the software library. MasterCard, Visa, COD.



PORTABLE
COMPUTER
SUPPORT
GROUP

1-214-351-0564

11035 Harry Hines Blvd., #207
Dallas, Texas 75229

PCSG offers a 30-day/money-back trial!

this instruction, for example, we could simply type in that text in this sequence:

```
-a
0930:0100 ADD AX,BX
0930:0102 MOV CX,23
0930:0104 ENTER
-
```

The A command after the DEBUG prompt tells DEBUG to *assemble* the following text into machine language instructions. In the example above, we have entered two instructions. Immediately after each, DEBUG has translated the text into the equivalent machine language code (which does *not* appear on the assembly). The first instruction ADD AX,BX is stored in locations 0930:0100 and 0930:0101. The second instruction MOV CX,23 is stored in locations 0930:0102 and 0930:0103. The ENTER terminates the assembly process and returns back to the DEBUG prompt. We can now use the U command to unassemble what we have just entered. The unassembly should match what we just entered (Example 2).

Note that in this case we specified the starting location for the unassembly in standard segment:offset form. We could also have used an offset alone:

```
-u 0100
```

As we had hoped, the first five bytes match the two instructions we had entered. (The remaining instructions reflect garbage bytes after the instructions which were not changed.) In addition, the machine language bytes assembled for the instructions appear in the second column after the locations of the instructions.

You can see from the unassembly that the ADD AX,BX in machine language form consists of two bytes, 01 and D8. The MOV CX,0023 consists of three bytes in machine language form, B9, 23 and 00. Note that the location counter is automatically incremented to point to the start of each new instruction.

That's basically the assembly process — entering text in the A mode and checking the text by a U command. The best way to do this, by the way, is to do a screen dump of assembly text, preferably after text has been entered. An alternative is to do a screen dump after unassembly. (Use the GRAPHICS command in MS-DOS before loading DEBUG and then press ENTER-SHIFT-PRINT on the Tandy 1000.)

Entering the text and assembling it is the easy part. The hard part is to know *which* mnemonics to enter and in what order to construct a program. The instruction set of the 8088 can be divided into logical groups, and we'll talk about one of those groups now.

Arithmetic Instructions

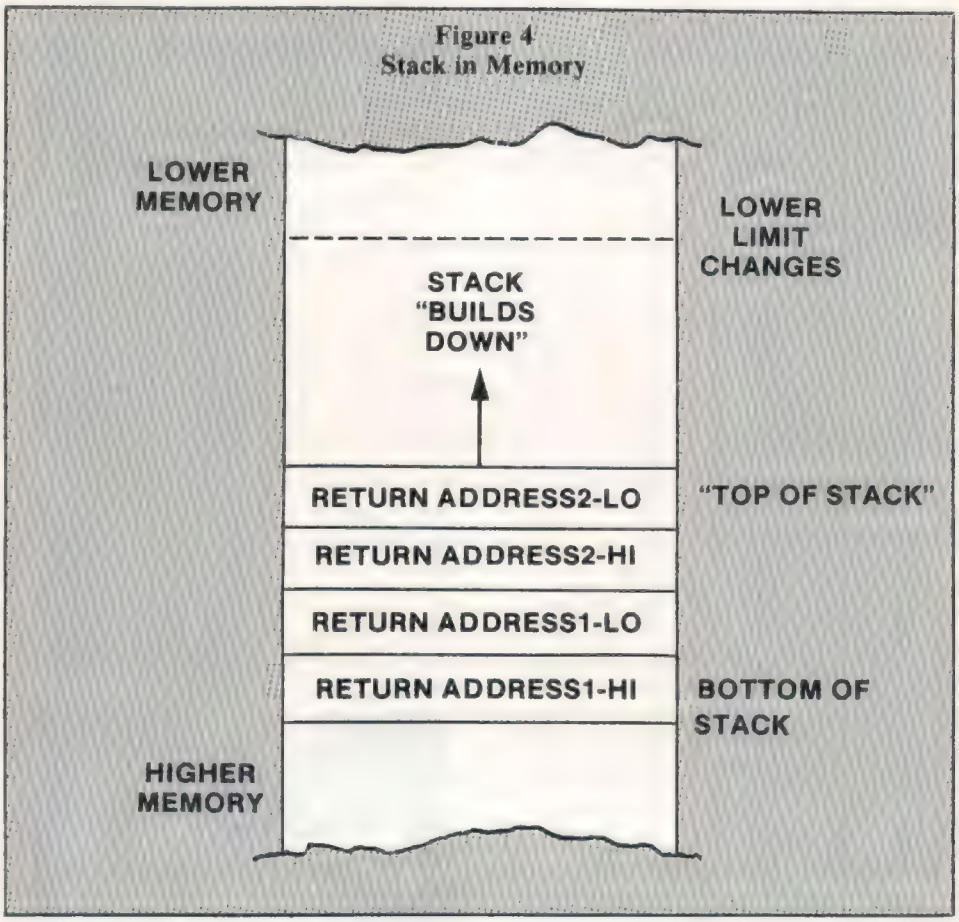
As you might expect, there's an ADD and SUB to add or subtract two operands in the 8088 instruction set. There's also a variation on the adds, ADC and SBB, which perform the same add or subtract, but also add in a "carry" or "borrow." INC and DEC increment or decrement (add or subtract one) from the contents of a register or memory location. A CMP compares two operands.

To see how some of these work, let's try a few examples. First of all, let's add the numbers from one to 10. Enter this sequence in DEBUG:

```
-a0100
0930:0100 mov ax,0
0930:0103 add ax,1
0930:0106 add ax,2
0930:0109 add ax,3
0930:010C add ax,4
0930:010F add ax,5
0930:0112 add ax,6
0930:0115 add ax,7
0930:0118 add ax,8
0930:011B add ax,9
0930:011E add ax,a
0930:0121 ENTER
-
```

What has been entered here is a complete assembly language program. It won't win any prizes in the "1985 Programmers Profundity" competition, but it's not bad for a start. Note

Figure 4
Stack in Memory



Example 2

```
-u 0930:0100
0930:0100 01D8          ADD     AX,BX
0930:0102 B92300        MOV     CX,0023
0930:0105 0000          ADD     [BX+SI],AL
-
-
```

that the last value to be added is a Hex A and not a 10, which would be Hex 10 or decimal 16. The first instruction *moves* an *immediate* value of zero into the AX register to clear it initially.

To execute the program we can use another DEBUG command, the G command, for GO. GO is similar to the BASIC GOTO. It executes from a given address. In this case, we'd like to execute starting at location 0930:0100 (or simply 0100 within the current segment). The format for this is (don't enter anything yet):

```
-g=0930:0100
```

or simply

```
-g=0100
```

One question arises, however. How does DEBUG know when to stop executing instructions? Wouldn't it just continue, trying to execute the 0000 byte as an ADD [BX+SI],AL instruction? It would, and for that reason, an ending point must be added in the G command. This ending point is called a *breakpoint*, the point at which DEBUG regains control. Before this breakpoint, DEBUG *has* lost control. Execution is done solely on a microprocessor basis without any interaction by DEBUG. The breakpoint substitutes a special instruction that returns control to DEBUG at the specified location. The location after the last instruction here is 0930:0121, and that's where the breakpoint should be. The format of the GO with the breakpoint is:

```
-g=0100 0121
```

If you execute this instruction, DEBUG will execute the instructions starting at 0100 and then display the registers after the breakpoint is reached (Example 3).

Note that the IP is now set to 0121 and that the current location on the third line of the display is 0930:0121 with a display of the "garbage" instruction at 0121. Looking in the AX register, we see 0037, a Hex number equivalent to decimal 55 or 1+2+3+4+5+6+7+8+9+10.

Now let's try another short program. This time we'll add the numbers from one through 100 in AX and use a *loop* to do it. Enter this code:

```
-a 0100
0930:0100 mov cx,64
0930:0103 mov ax,0
0930:0106 add ax,cx
0930:0108 dec cx
0930:0109 jnz 0106
0930:010B ENTER
-
```

This program introduces two new instructions, the DEC and JNZ. The DEC instruction is an arithmetic instruction we mentioned briefly above. It decrements one count from the contents of the A Register or memory location. In this case one is subtracted from CX. The JNZ is a *conditional* jump instruction, similar to a BASIC/IF/THEN statement. It causes a jump to the specified address if the count in CX is not zero, in this example.

Execute the program by

```
-g=0100 010B
```

After the breakpoint at 0930:010B is reached, you'll see the display in Example 4.

The AX register contains Hex 13BA, equivalent to 4096+ 768 + 176 + 10, or 5050, the result of adding the numbers from one through 100.

Conditional and Unconditional Jumps

The conditional jump instruction JNZ above needs some further explanation, as does setting the *flags* in the 8088. Arithmetic and other instructions such as ADDs and SUBs may have negative, positive or zero results, along with other conditions produced by the arithmetic operation. After an arithmetic instruction and certain other types of instructions are executed, a set of flags are set to the results of the operation. Refer to Figure 5 to see the flags. The nine flags are grouped together in a register to make them easier to handle, but, in fact, they are really nine separate flags.

The Z flag is easiest to understand. It is set or reset to reflect the zero or non-zero result of an arithmetic operation. If 23 is subtracted from 23, for example, the ZF flag would be a one. If 23 is subtracted from 24, the ZF flag would be a zero. Try these two instructions after assembling them:

```
-a 0100
0930:0100 mov ax,17
0930:0103 sub ax,17
0930:0103 ENTER
-g=100 106
```

After the breakpoint point is reached, you'll notice a string of two-letter mnemonics on the second line — NV UP DI PL ZR NA PE NC. These mnemonics correspond to the flags of the 8088 after the two instructions have been executed. The ZR mnemonic indicates the ZF flag is set and a zero result has occurred. Now reenter the sub ax,17, but change the 17 to 18. After executing and breakpointing again, you'll see an NZ in place of the ZR, indicating a *non-zero* result.

The other flags indicate other conditions set after specific instructions

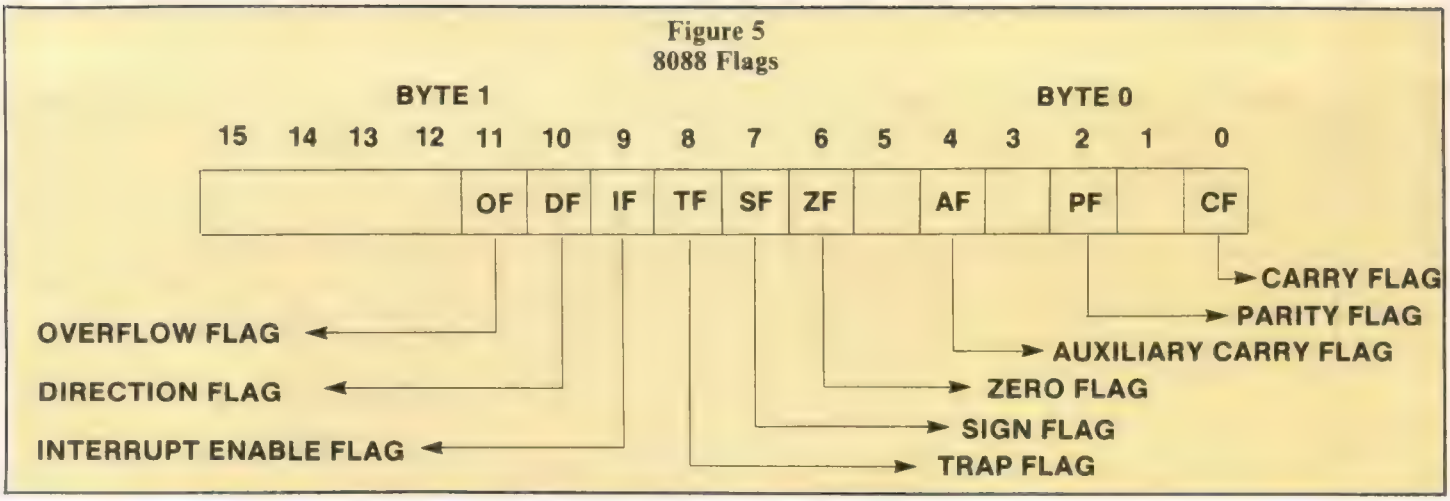
Example 3

```
AX=0037 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0930 ES=0930 SS=0930 CS=0930 IP=0121 NV UP DI PL NZ NA PO NC
0930:0121 0000        ADD     [BX+SI],AL                      DS:0000=CD
```

Example 4

```
AX=13BA BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0930 ES=0930 SS=0930 CS=0930 IP=010B NV UP DI PL ZR NA PO NC
0930:010B 0005        ADD     [DI],AL                         DS:0000=CD
```

Figure 5
8088 Flags



(usually arithmetic instructions). See Example 5.

Some of the flags are seldom used in programs, such as the parity flag or the auxiliary carry flag. The three most important flags in assembly language processing are probably the zero, sign and carry flags, in that order. We've discussed the zero flag above, so let's discuss the sign and carry flags.

The sign flag is set on the basis of the sign bit in a result. Numbers in the 8088 are either *absolute* or *signed* numbers. Absolute numbers have no sign — they are always positive. A good example of absolute numbers are numbers used to address memory. There are no negative memory locations, and hence only positive values are used.

Signed numbers, though, are used to express both negative and positive numbers. This type of representation is also called "two's complement" notation. In this format, numbers have a sign bit that is set (one) when the number is negative and reset (zero) when the number is positive. The format is shown in Figure 6. To see the form of positive and negative numbers, assemble this code:

```
mov ax,1
mov bx,-1
mov cx,1000
mov dx,-1000
```

Example 6

```
0930:0100 B80100    MOV   AX,0001
0930:0103 BBFFFF    MOV   BX,FFFF
0930:0106 B91000    MOV   CX,1000
0930:0109 BA00F0    MOV   DX,F000
```

If you entered this code and then unassembled, you'd see the display in Example 6.

The second and third bytes of each of the machine language codes represent the immediate data values to be moved into the register. They are in standard 8088 format, *least significant byte followed by most significant byte*, so we have to swap the two bytes to get the actual result. The first value is 0001, the second FFFF, the third 1000 and the fourth is F000. There's no problem on the 0001 and 1000 values — they appear as we would expect. However, the -1 and -1000 are in two's complement form, with the most significant bit set to a one for negative values.

To change from a two's complement form to an absolute form, do this: Change all ones to zeros and all zeros to ones and add one. In the case of the F000 value, for example, 1111000000000000 becomes 0000 111111111111. Adding one makes 0001000000000000, which is Hex 1000, the positive form of the number we had specified in the load.

To see how the sign flag is affected, try the following code:

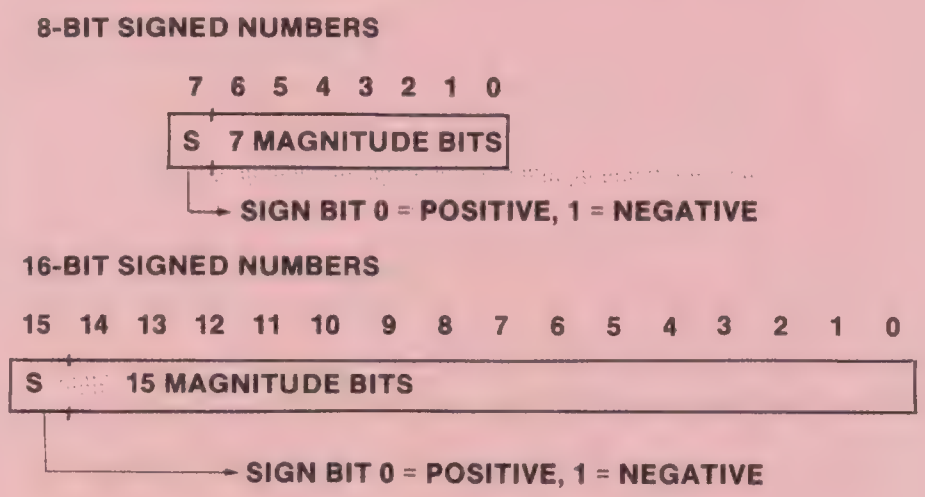
```
-a 0100
0930:0100 mov ax,17
0930:0103 sub ax,[109]
0930:0107 jmp 107
0930:0109 dw 23
0930:010B ENTER
```

This code introduces a few new things that we haven't discussed previously. The JMP instruction is similar to the one used previously, except that it is

Example 5

Flag	Explanation	Set Mnemonic	Reset Mnemonic
OF	Overflow flag — set if number too large to be held in register	OV	NV
DF	Direction flag — determines direction for certain string instructions	DN	UP
IF	Interrupt enable/disable	EI	DI
SF	Sign flag — set for negative results, reset for positive results	NG	PL
ZF	Zero flag — set if result is zero, reset if not	ZR	NZ
AF	Auxiliary carry — represents carry out of bit position 3	AC	NA
PF	Parity flag — count of number of one bits in result (even or odd)	PE	PO
CF	Carry flag — represents carry out of high-order bit	CY	NC

Figure 6
Signed (Two's Complement) Format



an *unconditional* jump — it always jumps to location 0107, that is, to itself. It's like a BASIC 100 GOTO 100 and put in only so we can breakpoint at 0107. The DW mnemonic is not for an instruction at all. It's called a "pseudo-op" as it's not really an opcode, but instructs the assembler to take some action, in this case putting in a constant of 23 in memory locations 0109 and 010A. The DW stands for "define word" and generates two bytes containing whatever value we've specified.

The last strange thing is the brackets around the 106 value in add,ax[106]. Why is this used? The brackets are used to signify a memory address rather than a literal value. If the brackets were not used in this case, the instruction would be an "add the value of 106 to the contents of the AX register." With brackets, it becomes an "add the contents of memory locations 0106/0107 to the contents of the AX register."

The operand to be added, therefore, is stored in a memory location. This type of *addressing mode* is called "memory operand to register" and is used often.

You can change the operand in 0930:0109 by doing an A function in DEBUG for 0109 alone:

```
-a 0109
0930:0109 _
```

Try executing from 0100 and breakpointing at location 0107 to see the results on the sign flag. In the case above, the result in AX will be 17-23 (or 23-35 decimal) or a value of -C (-12 decimal). You should see FFF4 in AX after the breakpoint and an NG for the sign flag, indicating that the result is negative. Try other values in the DW and observe the zero flag, sign flag and the results in AX.

The carry flag can be illustrated by a similar set of code. The carry flag is set when there is a carry out of the high order bit of a result. The carry we're talking about here is the same type of carry that occurs when you add two decimal numbers:

```
  9999
   +23
-------
1 ← 1022
```

The '1' above is a carry to the next digit position. If the decimal operation could hold only four digits (values of 0000 through 9999), we'd want to detect a carry to indicate that the result was too large, or perhaps to add to the next digit position for another add operation. The binary carry in assembly language is used for the same functions. Try assembling this code and watch the carry flag:

```
mov ax,FFFF
add ax,1
```

You should see the carry flag set to a CY, reflecting the fact that there was a carry out of bit position 15, the highest order bit of the AX register.

Another example to try is one that affects the overflow flag. The overflow flag is set when the result is too large to be held in eight or 16 bits as a *signed* value. For example:

```
mov ax,7fff
add ax,1
```

When this code is assembled and breakpointed, it will show the overflow flag set by OV. The reason for this is that the value 07FF is the maximum positive signed number that can be held in 16 bits. (Don't forget the sign bit — it must be a zero.) Adding one to this value produces a result of 8000 which is equal to -32,768 in two's complement or signed form, not at all what we'd like to see.

The flags are used all the time for conditional branches. We've seen an example of one, a conditional branch on the zero flag — JNZ — which branches if the result is non-zero and "falls through" if the result is zero, executing the instruction after the JNZ.

Most of the other flags can also be tested and used in conditional jumps as well. In place of the JNZ, we could use:

JZ	Jump if result zero
JE	Jump if result equal
JNE	Jump if result not equal
JS	Jump on sign (Jump if result negative)
JNS	Jump on no sign (Jump if result positive)

JC	Jump on carry
JNC	Jump on no carry
JO	Jump on overflow
JNO	Jump on no overflow
JP	Jump on parity even
JO	Jump on parity odd

Compare Instructions

We discussed compares briefly above. They are a special type of subtract which change only the flags and do not change operands in registers or memory. Compares are used in conjunction with a conditional jump such as the ones above. They are often used to test one operand against another for a "less than," "less than or equal," "greater than" or "greater than or equal condition." When this is done, additional conditional jump instructions are used. There are two sets of these, however — one for compares of absolute numbers and one for compares of signed numbers. The compares for absolute numbers use these conditional jumps:

JB	Jump if first operand less than (below) second
JBE	Jump if first operand less than or equal second
JAE	Jump if first operand greater than (above) or equal to second
JA	ump if first operand greater than second

The compares for signed numbers use these conditional jumps:

JL	Jump if first operand less than second
JLE	Jump if first operand less than or equal to second
JGE	Jump if first operand greater than or equal to second
JG	Jump if first operand greater than second

To see how these operations work, try assembling this code segment:

```
-a100
0930:0100 mov ax,[113]
0930:0103 cmp ax,[115]
0930:0107 je 10d
0930:0109 jl 10f
0930:010b jmp 111
0930:010d jmp 10d
0930:010f jmp 10f
0930:0111 jmp 111
0930:0113 dw 1000,2000
0930:0117 ENTER
-
```

You can now use the multiple breakpoint feature of DEBUG to breakpoint at 10d, 10f and 111. The first location is reached if the result of the compare is zero, the second is reached if the result is less than, and the third is reached if the result is greater than. Change the two operands for the compare in locations 113/114 and 115/116 — the ones in the code above are just samples of operands that could be put in. Also change the conditional jump instructions to see the effects of both absolute and signed numbers. The G0 command for the code above is

```
-g=100 10d 10f 111
```

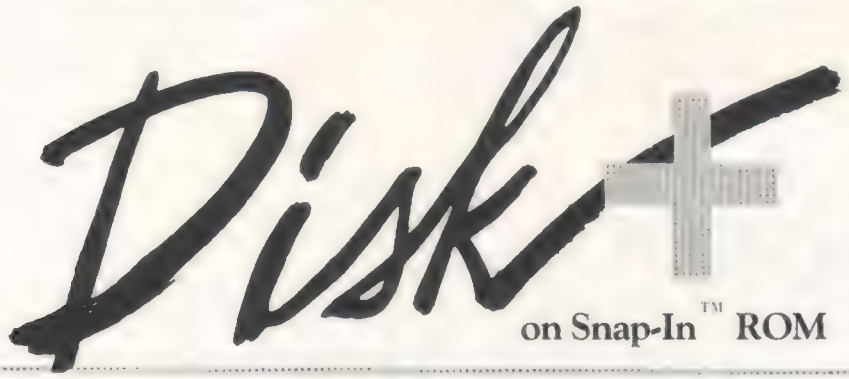
You'll see which breakpoint is reached by the contents of the IP and the location counter on the third line of the breakpoint.

Recap

We've covered a lot of ground in this lesson and you might well take the time between this column and the next to make certain that you understand what's been done above. However, if you've grasped the concepts presented here, you're well on your way to becoming an assembly language programmer. Next month we'll expand upon this basic material and show you how to construct simple programs and interface them to BASIC with DEBUG.

PCM

USE YOUR DESKTOP COMPUTER AS A DISK DRIVE FOR YOUR M-100.



Try *Disk+* for 30 days. If you aren't as excited as we are, return it for a full refund.

When we designed *Disk+* we did it out of necessity. We wanted a way that we could just connect a Model 100 to our desktop computer with a cable and save files onto the desktop's disk drive. We wanted it to be so simple to use it would be self-explanatory.

Picture this. *Disk+* comes to you on a Snap-in ROM and a diskette for your desktop. You take a quarter and open the little compartment on the back of your Model 100. Then you just press the ROM into the socket. *Disk+* appears on your main menu just like a built-in.

You connect your Model 100 to your other computer using an RS232 cable (available from PCSG for $40).

You just place the *Disk+* diskette into the desktop's drive and turn on the computer. It powers up automatically and says "awaiting command" on your desktop's screen. Then you just put the widebar cursor on the Model 100 main menu on *Disk+* and press ENTER. You are shown your RAM files arranged just like the main menu.

To save a file to your other system's disk drive, you just move the widebar cursor to the file you want to save and press ENTER. It is saved instantly with no further action.

To look at the disk directory, you just press a function key on your Model 100. You see immediately the disk directory on your Model 100 screen, and it is arranged just like your Model 100's main menu.

To load a file from the diskette to your Model 100, you just move the widebar cursor to the file and press ENTER. The file is transferred to your Model 100's RAM instantly. You can press F8 and go back to the main menu, and the file you loaded from diskette is there, ready to use.

It is so nice to be able to keep your documents, programs (both BASIC and machine code) and *Lucid* spreadsheet files on the diskette, and bring them back when you need them. All files are ready to run or use with no changes or protocol by you.

If you have access to a desktop computer and don't have *Disk+*, then evidently we have done a poor job telling you about it.

All files and programs that you load or save, go over and come back exactly as they are supposed to be because of full error checking. This guaranteed integrity is really a comfort. *Disk+* is wonderful in so many other ways. For example, you can do a "save all" of all your RAM files with just a touch of a function key. That group of files is saved on the diskette under a single filename with a .SD (for subdirectory) extension. Any time you want, you can bring back all those files at once, or just one or two if you like, again with one-button ease.

Disk+ takes up no RAM. That's zero bytes either for storing the program or for operating overhead.

What really excites most *Disk+* users is text file cross compatibility. Your Model 100's text files are usable on your desktop computer, and your desktop's text files become Model 100 text files.

This means you can write something on your Model 100, and with *Disk+* transfer it instantly to your desktop and start using it right away on your bigger computer. Or the way we like to work is to type in a document on the desktop computer and then transfer it to our Model 100 with *Disk+*. Then we print out the document, beautifully formatted, using WRITE ROM.

Disk+ works with just about every micro sold, from IBM PC and its clones, to all Radio Shack computers (yes, all), to Apple II, Kaypro, Epson and most CPM. Just ask us. More than likely, your computer is supported.

Incidentally, hundreds of Model 100 owners have gone to their Radio Shack stores and bought a color computer because it is so low priced, and with *Disk+* they have an inexpensive disk drive.

And if that weren't enough, how about this: *Disk+* also provides cross-compatibility between different computers like IBM, Apple or the Model 4 using the Model 100 as the intermediary device. Quite a feature!

The snap-in ROM is really great because you can use other ROMs like *Lucid* or WRITE ROM. They snap in and out as easily as an Atari game cartridge and you never lose your files in RAM.

Anyone who ever uses *Disk+* simply can't do without it. But so many times we have had new users call us and say, "Wow! I had no idea when I ordered it that *Disk+* would be so fantastic. I just couldn't believe that I could use my desktop computer's disk drive with my Model 100 just like it is another main menu."

That's why we sell *Disk+* on a thirty-day trial. If you aren't completely satisfied, return it within thirty days for a full refund. Priced at $149.95 on Snap-in ROM. MasterCard, Visa or COD.

1-214-351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd., #207, Dallas, Texas 75229 • 1(214) 351-0564

PCSG provides hotline software support for the Model 100. Call us at 1-214-351-0564

Take your portable copilot along with you on your next vacation or business trip

Portable Trip Companion

By Nathaniel F. Ireland

Anyone who has driven a trip of even moderate distance knows how difficult it is to drive and read a map. Fortunate indeed are those whose copilot is a skilled navigator; I am not so blessed. Years ago, I learned to sit down in the evening and make notes of the next day's routes and route changes, the start and end points, the approximate mileage and the general compass direction of each new route. With the help of a properly adjusted car compass, these notes have saved me countless hours of frustration and many wasted miles. Even my copilot found navigating easy . . . when she could read my writing.

Using the Model 100 and PLAN-R.BA, the task of creating such a trip plan has been made much easier. In fact, the whole trip plan textfile can be created in advance of the start of the trip and a printed copy obtained by using a suitable 40-column printer. While traveling, the trip plan textfile can be viewed directly on the Model 100 screen by a copilot.

Creating the TPLAN.DO Textfile

PLAN-R.BA is completely menu-driven and all entries are protected against destructive errors. Naturally, typing errors can still be made but these are easily edited in the textfile later. Some areas may cause confusion at first. On every map there is a statement that reads, "one inch equals approximately *xx* miles." This *xx* figure must be entered when the program calls for MAP MILES/INCH. Note that when this figure is entered, the new figure replaces the old figure and the typed figure disappears. Check the new figure for correctness and press ENTER if it is OK. If it is not correct, enter the corrected figure as above. Each FROM:TO: route entry will ask you to verify this figure as it is apt to change from map to map. Press ENTER if it remains the same, make the change as above if it is different.

The FROM: and TO: entries are program limited to five characters, VIA: entry limited to four. DIRECTION: is limited to one, either N,E,S or W.

INCHES: entry refers to the length in inches (plus any decimal part of an inch) of the route actually measured on the map with an ordinary ruler or a mechanical measuring device. For convenience, miles (plus any decimal part of mile) can be entered here if the right-most character typed is the letter M. Be careful to always include the letter M on the right when entering in miles because an error will be introduced in the day's accumulated miles count without it.

Trip Planning Suggestion

FROM: generally refers to the point at which you enter upon the route you indicate in VIA: and TO: generally indicates the point at which you leave that route. An exception to this is where the route continues onto another map. Since the new map is likely to have a different miles-per-inch figure and a state boundary line will be indicated on

---

(Nathaniel Ireland, now a retired gentleman farmer, was an engineer in the electronics industry for many years. In addition to his agricultural hobby, he finds time to enjoy his computers and do some consulting work.)

both maps, indicate the state line as TO: and measure to this point. Change the MAP MILES/INCH to agree with the new map, indicate the state line for the next FROM: and measure from this point.

Appending Data

Before creating a new TPLAN.DO textfile, PLAN-R.BA tests for an existing TPLAN.DO textfile in RAM. If it finds one, you are given a choice of adding to the existing file (append) or instructions for creating a new file. Before you elect to append, press M to go to the Model 100 menu and view the existing TPLAN.DO file. Note the number of the last day recorded. Add one to this number and enter it when asked for STARTING DAY NUMBER when you rerun PLAN-R.BA.

Program Operation Suggestion

Should you end the program before completing a day and wish to complete that day, delete DAY #x and all the entries for that day from the textfile. Rerun PLAN-R.BA, select the append option and re-enter all the day data.

Programming Note

In lines 140 and 330, do not omit the space after the first quote mark because this space directs program flow back to the start of the line when there is no keyboard input. The last GOTO statement directs the program flow back to the start of the line with any keyboard input other than the characters listed between the double quote marks.

Some Abbreviations I Use

I = Interstate Route
F = U.S. Route
S = State Route
LR = Local Road
KOA = Kampground Of America
SL = State Line
SPC = State Park Campground
NPC = National Park Campground
PCG = Private Campground □

PCM BAR CODED LISTING

The listing:

```
10 MAXFILES=1
20 REM ** PLAN-R.BA **
30 REM by N. F. Ireland, Jan 1984
40 REM modified May 1985
50 ONERRORGOTO170
100 REM SET UP RAM FILE
110 OPEN"RAM:TPLAN.DO"FORINPUTAS1
120 CLOSE:CLS:PRINT" TPLAN.DO already ex
ists in RAM. If you wish to start a new
 trip plan, press <M>, (1) save TPLAN
.DO to tape, & KILL TPLAN.DO in RAM OR
(2) rename TPLAN.DO using the NAME func
tion.
130 PRINT:PRINT" If you wish to append t
o the present TPLAN.DO in RAM, press
<A>.";
140 ONINSTR(" MmAa",INKEY$)GOTO140,150,1
50,160,160:GOTO140
150 MAXFILES=0:MENU
160 OPEN"TPLAN.DO"FORAPPENDAS1:GOTO210
170 IFERR=52THENCLOSE:OPEN"RAM:TPLAN.DO"
FOROUTPUTAS1:RESUME180:ELSEPRINT"ERROR C
ODE";ERR;"IN LINE ";ERL:STOP
180 CLS:PRINT" PLAN-R TRIP PLANNIN
G PROGRAM":PRINT" by N. F. Ir
eland"
185 PRINT@120,;:LINEINPUT"ENTER TRIP TIT
LE:";TI$:IFLEN(TI$)>39THENPRINT@120,SPAC
E$(79):GOTO185
190 PRINT@200,;:LINEINPUT"ENTER DATE OF
TRIP:";DA$:D=1:IFLEN(DA$)>39THENPRINT@20
0,SPACE$(79):GOTO190
195 LE=LEN(TI$):P=INT(20-LE/2):TI$=SPACE
$(P)+TI$:LE=LEN(DA$):P=INT(20-LE/2):DA$=
SPACE$(P)+DA$:PRINT#1,TI$:PRINT#1,DA$:PR
INT#1,SPACE$(39)
200 REM BUILD TRIP PLAN
210 CLS:IFD=0THENINPUT"ENTER DAY NO.:";D
:ELSEPRINT"DAY NO. ";D
220 PRINT#1,"DAY #";D:GOTO240
230 CLS:PRINT"FROM:";F$;" TO:";T$;" VIA:
";V$;" D:";D$;" M:";:PRINTUSING"###.#";M
240 PRINT@40,"MAP MILES/INCH=";MI;:LINEI
NPUTMI$:IFMI$<>""THENMI=VAL(MI$):MI$="":
PRINT@40,SPACE$(39):GOTO240
250 PRINT@80,;:LINEINPUT"FROM:";F$:IFLEN
(F$)>5ORLEN(F$)<1THENPRINT@80,SPACE$(39)
:GOTO250:ELSEIFLEN(F$)<5THENLE=LEN(F$):P
=5-LE:F$=F$+SPACE$(P)
260 PRINT@120,;:LINEINPUT"TO:";T$:IFLEN(
T$)>5ORLEN(T$)<1THENPRINT@120,SPACE$(39)
:GOTO260:ELSEIFLEN(T$)<5THENLE=LEN(T$):P
=5-LE:T$=T$+SPACE$(P)
270 PRINT@160,;:LINEINPUT"VIA:";V$:IFLEN
(V$)>4ORLEN(V$)<1THENPRINT@160,SPACE$(39
):GOTO270:ELSEIFLEN(V$)<4THENLE=LEN(V$):
P=4-LE:V$=V$+SPACE$(P)
280 PRINT@200,;:LINEINPUT"DIRECTION:";D$
:S=INSTR("NnEeSsWw",D$):IFLEN(D$)<>1ORS=
0THENPRINT@200,SPACE$(39):GOTO280
290 PRINT@240,;:LINEINPUT"INCHES:";I$:IF
VAL(I$)=0THENPRINT@240,SPACE$(39):GOTO29
0:ELSEIFRIGHT$(I$,1)="M"ORRIGHT$(I$,1)="
m"THENM=VAL(I$):I$="":ELSEM=MI*VAL(I$):I
$=""
295 TM=TM+M:PRINT@258,"ACCUM. MI.=";:PRI
NTUSING"###.#";TM
300 REM RECORD DAY INFO TO RAM
310 PRINT#1,"FROM:";F$;" TO:";T$;" ON:";
V$;" D:"D$" M:";:PRINT#1,USING"###.#";M
320 PRINT"Press <M>ore, <N>ext day, <E>n
d prog.";
330 ONINSTR(" MmNnEe",INKEY$)GOTO330,230
,230,340,340,350,350:GOTO330
340 PRINT#1,SPACE$(10)+"TOTAL DAY MILEAG
E ";:PRINT#1,USING"###.#";TM:PRINT#1,SPA
CE$(39):TM=0:D=D+1:GOTO210
350 PRINT#1,SPACE$(10)+"TOTAL DAY MILEAG
E ";:PRINT#1,USING"###.#";TM:CLOSE:MAXFI
LES=0:MENU
```

PCM

YOU CHOOSE:

	MLINK	Crosstalk XVI	Smartcom II
Terminal Emulation Mode			
Menu-driven Interface	Yes	Yes	Yes
Expert/brief Command Mode	Yes	Yes	
Extensive Help Facility	Yes	Yes	Yes
Directory-based Autodialing	Yes		
Automatic Logon	Yes	Yes	Yes
Programmable Function Keys	Yes	Yes	
Multiple Modem Support	Yes	Yes	
File Transfer Mode			
Error Checking Protocol	Yes	Yes	Yes
Wildcard File Transfers	Yes	Yes	Yes
File Transfer Lists	Yes		
XMODEM Protocol Support	Yes	Yes	Yes
Mainframe Versions Available	Yes		
Command Language			
Conditional Instructions	Yes	Yes	
User Variables	Yes		
Labels	Yes	Yes	
Fast Interpreted Object Code	Yes		
Program Run	Yes	Yes	
Subroutines	Yes		
Arithmetic and String Instructions	Yes		
Debugger	Yes		
Miscellaneous			
Remote Access	Yes	Yes	Yes
Electronic Mail	Yes		
Unattended Scheduling	Yes	Yes	
Expandable Interface	Yes		
CP/M Versions Available	Yes		

MLINK™

The choice is easy. Our MLINK Data Communications System is the most powerful and flexible telecommunications software you can buy for your personal computer. And it's easy to use. MLINK comes complete with all of the features listed above, a clear and comprehensive 275-page manual, and 21 applications scripts which show you how our unique script language satisfies the most demanding requirements.

Choose the best. Choose MLINK.

Corporate Microsystems, Inc.
P.O. Box 277
Etna, NH 03750
(603) 448-5193

Introductory offer - we pay shipping!
Include $250.00 for each copy of MLINK.
Outside North America add $15.00 (U.S.)

___ Check enclosed ___ MasterCard ___ Visa
___ COD (Registered checks or money orders only)

Please send me the following version(s) of MLINK:

_____ Tandy Model 1000/1200 (MS-DOS)
_____ Tandy Model 2000 (MS-DOS)
_____ Tandy Model 4 or 4P (CP/M Plus only)
_____ IBM® PC or compatible (PC-DOS)
_____ Send me more information

Other versions are also available.
Please call for current version information.

Card # _____ Exp. _____

Signature _____

Ship to:
Name _____

Address _____

City/State/Zip _____

Subsequent Thought on Graphics Hunting

By Randy Graham
PCM Contributing Editor

Last time, we talked about finding graphics files in online databases. I'd like to say a couple of more things on that general subject. For one, I want to answer the question no one is asking: Why do we turn to a column about telecommunications to read an article about a "local" application like graphics? Well, for two reasons.

I thought some of you may be like me — totally lacking in artistic ability. Ever since I've had a computer, I wanted to get into graphics, hoping the machine would be smarter than me. I cannot draw a straight line without a ruler or a circle without a compass, but perhaps with a graphics program, I can create great art for my own enjoyment and perhaps some practical uses.

But, alas, talent is still required unless I am given a box full of pieces to paste together. My last hope is to find graphics files that I can use, and modify them to suit my particular requirements. If this sounds like your lament, too, you may be willing to search online as I have for usable pretty pictures.

My other thought was in the category of "reinventing the wheel." Creating good graphics, like programming, can be a tedious process. Perhaps you can do it but do not have the time. Perhaps someone has already done the work and will share it with you, or maybe you have done the work and do not want to simply put it on the shelf.

I am still amazed at the small proportion of computer users who take advantage of the telecommunicating power of their equipment. I keep looking for examples to enlarge your horizons and persuade you to take the plunge.

Another Kind of Graphics

Last month we talked mostly about Hi-Res screen graphics, just noting that with a screen dump program you can print them out for reproduction. Let's say another word about print graphics — pictures, illustrations, charts, graphs, maps, etc. Again, this is a largely "local" activity and the print utility must be specific to the computer and the printer. With a little luck and successful searching, you may find files online that can be used.

Print graphics have one general characteristic — they run slowly. Remember how the graphics programs read or write to the screen display area of memory? The simplest and crudest just use a pixel set/reset approach, so that shades of gray depend on density of dots. More sophisticated programs include data on brightness of the pixel or even its color. These pixels are set in horizontal rows as the CRT scans each line. In print graphics, the image is generated by the column of wires in the dot-matrix printhead, and the printer wants to print seven, eight or nine rows at a time as it moves across the page. The utility must, therefore, set up an appropriate-sized array, read each element of that array, digitize it and send it to the printer to set or reset the print wires.

The effect of this cumbersome process is that there is a noticeable pause between the printing of each row of a picture. For online use, this makes it important that you download the file, save it, get offline and do your printing. Otherwise, you will pay for a lot of connect time while waiting for the printer to sort things out.

And Another Kind of Analog

Last month (one more time), to introduce the subject of graphics, I talked a little about the digitizing process. This works fine for pictures that are generated for screen or printer display. There is another artistic application that converts continuously variable values to digital form for manipulation and storage — music. Most computers can "sing" and programmers have produced utilities that convert notes to sound. The best of them have very good quality, but let's face it, it always sounds like the synthesized music it is.

My calculus teacher finally persuaded me that a curve is a series of discrete points where the increment of change is so small it is perceived as

---

(Randy Graham is a rehabilitation counselor working with the handicapped. Personal computing is his hobby; telecommunications, one of his favorite activities. He has done freelance information retrieval and is an inveterate user of the major online systems.)

continuous. But, when the variables are read and the values are digitized, memory requirements get to be horrendous. A compromise is to sample the curve in time, producing a series of steps.

In graphics, a curve becomes a staircase when the time between samples is large enough so the file will fit into available memory. In computer music, this staircase affects all aspects of the music's quality and makes it sound "electronic."

Nevertheless, for us unartistic types, it is a way to make our equipment "sing." Again, doing your own can be a time-consuming process; you may like to sample the results of others' work in online libraries. Like graphics, such libraries usually have the necessary utility available. Give them a listen. Just make sure the music program runs on your machine.

News off the Wire

While you and I were off on vacation, dangling our toes in the creek or whatever, the information services were busy trying to become more user friendly. They are facing up to the problems of the occasional user. The professional who logs on every day soon becomes so familiar with a system's protocols that he/she almost never has to consult a manual. When you just check in from time to time, little differences can cause a lot of frustration and wasted time. They have heard us and they are trying to help.

Dow Jones has implemented a new, simpler menu and charge system that is much easier to understand. CompuServe has improved its "navigating" process, so that, among other things, it "talks" to you more naturally and "understands" an easier-to-learn command structure.

Perhaps the most ambitious get-acquainted program is The Source's new *TUTOR* program. It is a well-written introduction to their services and it is free to all members — you are not charged for connect time while running *TUTOR*. I tried it out and found it easy to use and informative. I recommend it highly to those who want to learn to use information services.

I am sorry I did not hear in time to tell you about Delphi's special free membership offer during the month of July. Delphi is still a good deal, and the promotion was to get the attention of Tandy owners. They have started a Tandy users' SIG; one section supports the 1000, 1200HD, 2000 and 6000. This is the first such SIG I have run across. There is another section for portables, but then "PoCophiles" already have our SIGs on CompuServe and The Source.

The quickest way to get more information about signing up for Delphi is to call them at 1-800-544-4005. Tell them you just read about their free offer in PCM and want to sign up. I don't think it will work, but what have you got to lose?

On to Princeton

As you read this sometime after Labor Day, there is time to sign up for the PCMfest in Princeton, N.J., Oct. 11-13. Get those tickets and reservations (see the ad in this issue). If you hurry, there may still be time to get a super-saver air fare. I attended RAINBOWfest at the Princeton Hyatt last year and can testify that it is a fabulous place for an interesting and exciting show. Let's all get together for our first annual PCM family reunion. Be on the lookout for my name tag — I will enjoy meeting you face to face! PCM

This menu utility makes BASIC programming easier

A VERY BASIC MENU

By Cecil M. Goeldner

I spend a lot of time in BASIC writing new programs and enhancing old ones. Often, however, my time is broken up in short increments for such minor things as getting some sleep, going to work, being called to dinner and other miscellaneous interruptions. Operating in this mode, I soon became acutely aware of the need for a method to quickly pick up the scent from previous sessions. In true programmer fashion, I responded to this requirement by writing a program. I call it *My BASIC Menu* and have included several functions that I find quite useful and real time-savers.

As I developed this program I decided the first priority would be to assign my own values to the soft keys. I felt the initial values I got each time I brought up BASIC were just not functionally productive and so I changed them to better suit my needs.

Referring to the program listing, lines 100-112 contain the entries I now use for this purpose. Preliminary steps are taken in Line 100 to clear all variables, clear the screen, set the screen width and turn on the soft key display on Line 25 at the bottom of the screen. Now I don't usually leave this soft key display on during an entire session, but I do prefer it on initially as a reminder of the current functions, especially since I tend to change them from time to time as I endeavor to improve their usefulness. Besides, it's always fun to try something new.

So far, my favorite soft key settings seem to center around the LIST command, and, in this case, I have given each key from F1 through F9, a unique line number range.

Since most of my programs are small, and I like to code complete routines using line number ranges from n00 to n99, a listing scheme like this just naturally follows; however, larger ranges could be specified if desired. If I am working in the middle of a routine and have a question about another segment of the program, I just press the soft key corresponding to the line number range containing the program segment in question, and it is immediately displayed. Because I generally code routines in small, easy to handle segments, they will usually fit on the screen without scrolling over.

It is fairly easy to remember the range of numbers for most routines as they are developed. As you can see in lines 101 through 109, I use the key number as the high order digit and just tack on two more digits for the range. Thus, soft key F1 will list program lines 1 through 199, F2 will list lines 200 through 299 and so on up through F8 which will correspondingly list line numbers 800 through 899. Notice that I include the CHR$(13) on the end of the string value settings for each of the keys F1 through F8. This is the carriage return code, and without it I would have to press the enter key each time following the soft key in order to get the function performed. However, for the F9 key I just left it as an open LIST command and that way I can enter the range manually or just press the enter key to list the entire program.

Soft key F10 is set with only the single character CHR$(12). This is the form feed code and accomplishes the same thing as a CLS (clear screen) except that it leaves the cursor located at the very top left-hand position of the cleared screen, whereas the CLS leaves the cursor positioned one line below. Since I like to use the entire screen, I find

---

(Cecil is a senior systems programmer for a large Michigan bank and has been actively involved in computer programming at all levels for more than 20 years.)

it useful to have the cursor situated at the top line of a clear screen and this will do it for me in one stroke. Additionally, I like to combine the key functions such as F10 followed by F5, which will clear the screen and list program segment lines 500 through 599.

Due to screen width size, the soft key display at the bottom of the screen shows only those values for keys F1 through F10, and yet there are twelve soft keys. How do you see the values for the remaining two keys — Numbers 11 and 12? You can toggle the display by pressing CTRL-T, but I don't usually bother. I assign functions to these two keys that I won't change or forget and, therefore, I don't need to see them. My unforgettable Number 11 is the RUN command bracketed on the left by a CHR$(12) and on the right by the carriage return CHR$(13). The CHR$(12) in this case does two things for me: First, it clears the screen before running the program under construction; and secondly, it permits the RUN command to continue without a bothersome syntax error if the cursor just happens to be located somewhere in the middle of a busy screen at the moment I decide to "give it a try." Otherwise, I have to manually clear the screen first by pressing CTRL-L or at least clear the line at the cursor location by pressing CTRL-E.

Figure 1. Screen display showing "My BASIC Menu" and prompt awaiting user response.

Figure 2. Screen display showing the result of requesting action step 1 in response to initial menu prompt.

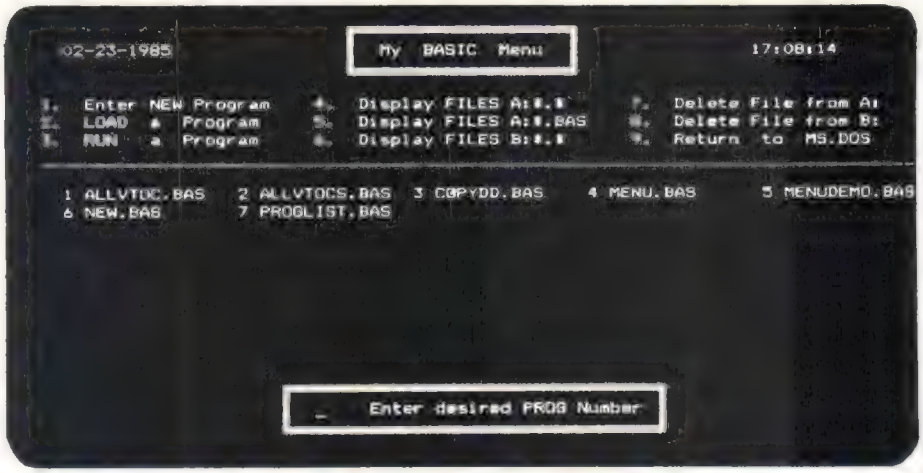


Figure 3. Screen display showing results of requesting action step 2 or 3 now waiting for user to select which program to LOAD or RUN.

This brings us to the last soft key, Number F12. I reserve it for the most important function of all, that of running *My BASIC Menu* program itself. There are, in fact, two methods of running this program which I employ quite frequently. One is by pressing soft key F12, and the second, undoubtedly the handiest, is to make an AUTOEXEC.BAT file which includes the BASIC command immediately followed by one space and the program name MENU right on the same line. This gets the whole thing started with the least amount of fuss and bother. I happen to use an AUTOEXEC.BAT file that looks something like this:

```
DATE
TIME
BASIC MENU
```

What could be simpler? Every time I turn on the system it will automatically boot up BASIC and execute *My BASIC Menu* without any intervention on my part, other than keying in the current date and time. Yes, I always enter these as they provide so many benefits in keeping track of files and things in an orderly fashion. I realize it may not always be convenient or even desirable to go through this exercise each time you turn on the system, and that is why I keep this AUTOEXEC.BAS file only on those diskettes which I intend to use for BASIC programming. It also comes in handy to review contents of other diskettes without leaving BASIC. This may not seem completely earth shaking now, but let us examine the various menu options to help bring it all in perspective on the Richter scale.

When the program is executed, "My BASIC Menu" is displayed, followed by the menu options (see Figure 1).

These menu options are for the most part self-explanatory, but they do deserve a brief explanation. I will describe each in turn, but in keeping

Figure 4. Typical screen display from Menu step numbers 4, 5 or 6. Note the filespec names are shown in ascending sequence.

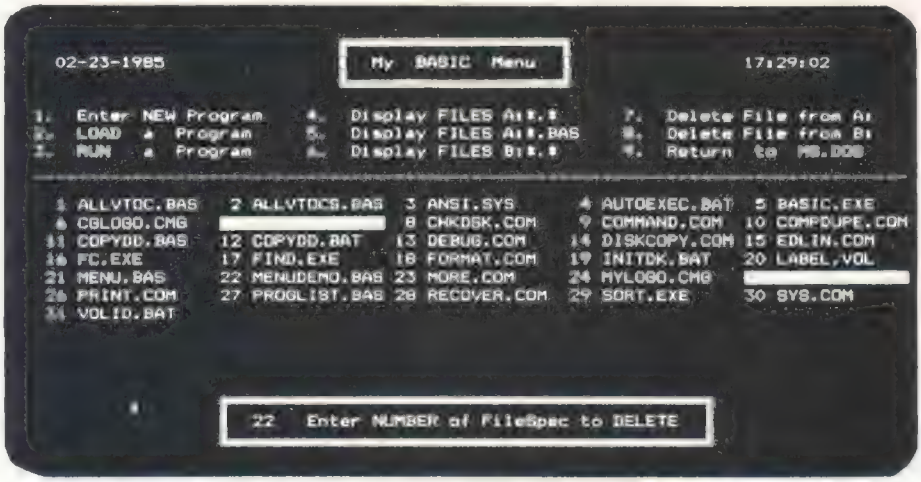


Figure 5. Screen display typical to action steps 7 or 8 showing two files (7 and 25) already deleted and file number 22 about to be deleted when the enter key is pressed.

with the flow of the program, I need to expand just a bit on the 200-290 and 300-380 line number ranges. The menu itself is presented to the user at lines 200-290. Each option display line value is put in the S$ string variable and the location where it is to be displayed on the screen is contained in the variables R% (row) and C% (column). The GOSUB 600 for each menu segment performs the common screen print subroutine. The single GOSUB 995 performs a subroutine to display the current date and time on the screen (another good reason for entering actual date and time at startup).

The 300 series line numbers requests the user to enter the desired step number and uses the ON . . . GOTO statement to pass control to the appropriate program segment corresponding to the MSN% (menu step number) which was entered. There are also some cosmetic instructions in this area; for example, Line 300 sets some values and issues a GOSUB 660 which results in the construction of the double line box that surrounds the menu name at the top of the screen. Similarly, Line 320 results in the single line box at the bottom of the screen used to highlight the prompt. Line 310 draws a horizontal line from the middle of the screen out toward both sides, which serves as a boundary line to visually separate the menu from subsequent displays resulting from various requested options (see Figure 1).

The listing:

```
100 CLS: CLEAR: KEY ON: WIDTH 80
101 KEY 1, "LIST 1-199"+CHR$(13)
102 KEY 2, "LIST 200-299"+CHR$(13)
103 KEY 3, "LIST 300-399"+CHR$(13)
104 KEY 4, "LIST 400-499"+CHR$(13)
105 KEY 5, "LIST 500-599"+CHR$(13)
106 KEY 6, "LIST 600-699"+CHR$(13)
107 KEY 7, "LIST 700-799"+CHR$(13)
108 KEY 8, "LIST 800-899"+CHR$(13)
109 KEY 9, "LIST "
110 KEY 10, CHR$(12)
111 KEY 11, CHR$(12)+"RUN"+CHR$(13)
112 KEY 12,"RUN "+CHR$(34)+"MENU"+CHR$(34)+CHR$(13)
200 R%=2:  C%=32: S$="My  BASIC  Menu":              GOSUB 600: GOSUB 995
210 R%=5:  C%=1:  S$="1.  Enter NEW Program":       GOSUB 600
220 R%=6:         S$="2.  LOAD  a  Program":        GOSUB 600
230 R%=7:         S$="3.  RUN   a  Program":        GOSUB 600
240 R%=5:  C%=26: S$="4.  Display FILES A:*.*":     GOSUB 600
250 R%=6:         S$="5.  Display FILES A:*.BAS":   GOSUB 600
260 R%=7:         S$="6.  Display FILES B:*.*":     GOSUB 600
270 R%=5:  C%=55: S$="7.  Delete File from A:":     GOSUB 600
280 R%=6:         S$="8.  Delete File from B:":     GOSUB 600
290 R%=7:         S$="9.  Return  to  MS.DOS":      GOSUB 600
```

Line 340 prompts the user for the desired option number and the GOSUB 500 executes the keyboard data entry subroutine to obtain it. Line 340 turns off the soft key function display at the bottom of the screen. Remember, I mentioned earlier that I only wanted it on briefly as a reminder, and as soon as an option has been selected I have elected to turn it off because it soon becomes distracting. The last two lines in the 300 range are used to verify the MSN% (menu step number), repeating the prompt if necessary and branching to the appropriate processing point otherwise.

Following the program logic sequentially, we arrive at Line 400 if MSN% equals one, indicating a desire to key in a new program. At this point, the screen is cleared and the words "New Program" are displayed centered in the top line. Then the disk file NEW.BAS is opened as output, and one record written to it containing the data 100 AUTO 100,10. This file is immediately closed. The RUN statement causes this file to be executed as a BASIC program, whereupon the MENU program is deleted from memory and is replaced by the new one. The automatic line numbering feature is activated, and the user can begin entering statements (see Figure

```
300 BOXR%=1: BOXC%=29: BOXW%=21: BOXH%=3: BLV%=2: GOSUB 660
310 FOR N%=0 TO 80 STEP 2: R%=8: C%=0: S$=STRING$(N%,95): GOSUB 600: NEXT N%
320 BOXR%=21: BOXC%=23: BOXW%=36: BOXH%=3: BLV%=1: GOSUB 660
330 R%=22: C%=26: S$="       Enter desired STEP Number": GOSUB 600: GOSUB 995
340 R%=22: C%=26: S$="": GOSUB 600: GOSUB 500: KEY OFF
360 MSN%=K%: IF MSN%<1 OR MSN%>9 OR LEN(K$)>1 THEN 300
380 ON MSN% GOTO 400, 420, 430, 440, 450, 460, 470, 480, 490
400 CLS: R%=1: C%=0: S$="New Program" + CHR$(13): GOSUB 600
410 OPEN "O",1,"NEW.BAS": PRINT# 1,"100 AUTO 100,10": CLOSE: RUN "NEW.BAS"
420 PNM$="A:*.BAS": PSW%=1: GOTO 700
430 PNM$="A:*.BAS": PSW%=1: GOTO 700
440 PNM$="A:*.*": PSW%=0: GOTO 700
450 PNM$="A:*.BAS": PSW%=0: GOTO 700
460 PNM$="B:*.*": PSW%=0: GOTO 700
470 PNM$="A:*.*": PSW%=1: GOTO 700
480 PNM$="B:*.*": PSW%=1: GOTO 700
490 CLS: CLEAR: SYSTEM: END
500 A$="": K$=INKEY$: K$="": RK%=R%: CK%=C%: LOCATE R%,C%,1: PRINT "_";
```

2). To turn off AUTO, press the BREAK key which cancels the current line and returns to BASIC command level.

As the result of requesting any of the menu steps two through eight, the corresponding lines 420 through 480 will be executed, which establishes the value of the string variable PNM$. This value is used later as the object of a FILES statement, which displays selected program names from the diskette. Menu step nine permits the user to exit BASIC and return to MS-DOS (Line 490).

The keyboard data entry subroutine begins at Line 500, and upon exiting at Line 550 the string variable K$ contains the character(s) just entered. If K$ contains no data (null string), then the program branches to Line 999 and promptly ends. The integer variable K% also contains the numeric value of K$ and is provided for those cases where the field entered is to be numeric. This eliminates any requirement for a separate numeric-only data entry subroutine.

The simple common screen print subroutine is made up of but two lines. At 600 the value of C% (column location)

```
510 WHILE K$="": GOSUB 995: K$=INKEY$: WEND
520 IF K$=CHR$(13) THEN 550
530 A$=A$+K$: S$=A$: R%=RK%: C%=CK%: GOSUB 600
540              GOSUB 995:  K$="":  GOTO  510
550 K$=A$: K%=VAL(K$):  GOSUB 995: IF K$="" THEN 999 ELSE RETURN
600 IF C%<1 THEN C%=(80-LEN(S$))/2: IF C%<1 THEN C%=1
620 LOCATE R%,C%,0:  PRINT S$;:  RETURN
660 '  (BOXR%, BOXC%=5) = UPPER LEFT CORNER:  BOXW%=WIDTH: BOXH%=HEIGHT
661 IF BLV%=1 THEN UL%=218: HL%=196: UR%=191: VL%=179: LL%=192: LR%=217
662 IF BLV%=2 THEN UL%=201: HL%=205: UR%=187: VL%=186: LL%=200: LR%=188
663 S$=CHR$(UL%)+STRING$(BOXW%-2,HL%)+CHR$(UR%)
664     R%=BOXR%: C%=BOXC%: GOSUB 600
665 IF BOXH%<2 THEN 668
666 FOR R%=BOXR%+1 TO BOXR%+BOXH%-2: S$=CHR$(VL%)
667     C%=BOXC%: GOSUB 600: C%=BOXC%+BOXW%-1: GOSUB 600: NEXT R%
668 S$=CHR$(LL%)+STRING$(BOXW%-2,HL%)+CHR$(LR%)
669     R%=BOXR%+BOXH%-1: C%=BOXC%: GOSUB 600:  RETURN
700 GOSUB 990: FILES PNM$: CC%=POS(C): CR%=CSRLIN
710 DA%=DA%+1: IF DA%>1 THEN ERASE PNF$
```

Factory Authorized

Radio Shack® DEALER

Discover a Whole New World of TV Entertainment!

SATELLITE SYSTEM

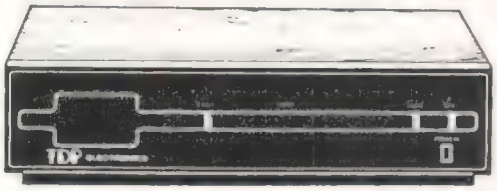
U.L. Approved

85° LNA

User Installable! Over 100 Channels! Uncut Movies! Specials! Sports! News! Weather! Music Videos!

Entire System **$1699**

- Totally Automatic System With 10½-Ft. Dish, Receiver and Wireless Remote Control
- Comes With All Accessories Needed for Complete Do-it-Yourself Installation
- Descrambler Ready



Handy Wireless Remote Lets You Change Channels, Control Volume and Fine-Tune

Now you can enjoy it all! 24-hour non-stop entertainment that you simply can't get from local TV, cable, and even video tape! You'll see the hottest stars, recent uncut and uninterrupted movies, the ultimate in sports coverage—and the best part is, there are no monthly fees to pay. You'll see exciting family-oriented programs, plus music videos, FM broadcasts and movies with provision for stereo, and 24-hour news coverage on both TV and radio. It all adds up to out-of-this-world TV viewing at a down-to-earth price. Come in today and discover how easy it is to enjoy satellite programming! Includes complete installation manual.

(817) 825-4027

NOCONA ELECTRONICS • Box 593 • Nocona, TX 76255

UPS Shipping

A RADIO SHACK DEALER

is tested, and if found to be less than one, a new value is computed which will result in centering the display field. At Line 620 the string variable S$ is displayed on the screen at location R% (row) and C% (column).

Lines 660 through 669 contain a unique little subroutine that I like to use to add a little something extra and to give those special screen displays that nice crisp, yet smooth, professional look. This routine draws boxes such as those mentioned earlier that enclose the menu name and the user prompt request.

Line 700 is the common entry point for menu steps two through eight once the value of PNM$ has been established (lines 420-480). At this time the FILES statement is used to display the requested filenames from the desired disk drive as specified in the respective menu line. Then the PNF$ (program name filespec) array is created by taking the names from the FILES screen display. This array is sorted (lines 800-820) and redisplayed in ascending sequence (lines 840-880). At this point (Line 890), menu steps four, five and six (see Figure 4) have been completed and program control is returned to Line 300 for any further selections.

Menu steps two and three functions are completed at lines 900 through 930 where the user selects from the sorted display, the filespec which is to be loaded or run (see Figure 3). These items are selected by keying in the sequence number located just to the left of the desired filespec rather than keying in the actual filespec name itself. Menu steps seven and eight delete functions which are wrapped up at lines 935 through 985. In similar fashion to that of options two and three, the user selects the number of the desired program filespec. The specified file is then deleted from the diskette, and the program name is blotted out from the display (see Figure 5). The user is then prompted for additional deletes which are subsequently handled in like manner. If there are none, the program returns to the main menu for further action.

Lines 990 and 995 are both one-line subroutines used for partial screen clear and date/time displays. The last line number, 999, is executed when the user responds to a prompt by pressing only the ENTER key. This ends the program, and control is returned to BASIC command mode. □

```
72Ø A%=(CR%-1Ø)*6 + INT((CC%+1)/13):  DIM PNF$(A%): A%=Ø
73Ø FOR R%=1Ø TO CR%:
74Ø  FOR X%=Ø TO 5:  P$="":  FOR C%=(13*X%+1) TO (13*X%+12)
75Ø   IF SCREEN(R%,C%)<>32 THEN P$=P$+CHR$(SCREEN(R%,C%))
76Ø   NEXT C%
77Ø   IF P$="" OR P$=STRING$(LEN(P$),32) THEN X%=5: R%=CR%: GOTO 79Ø
78Ø   A%=A%+1:  PNF$(A%)=P$
79Ø   NEXT X%: NEXT R%: IF A%<2 THEN 89Ø
8ØØ  FOR S%=1 TO A%-1:  IF NOT PNF$(S%)>PNF$(S%+1) THEN 82Ø
81Ø   SWAP PNF$(S%),PNF$(S%+1):  S%=S%-2: IF S%<1 THEN S%=Ø
82Ø   NEXT S%:  GOSUB 99Ø
84Ø  FOR I%=1 TO A%: R%=INT((I%-1)/5): C%=(I%-(R%*5)-1)*16+1
85Ø   IF PSW%=Ø THEN LOCATE R%+1Ø,C%+2:   PRINT PNF$(I%);:  GOTO 87Ø
86Ø   LOCATE R%+1Ø,C%: PRINT USING "###"; I%;: PRINT CHR$(32); PNF$(I%);
87Ø   GOSUB 995
88Ø   NEXT I%
89Ø ON MSN% GOTO 41Ø, 9ØØ, 9ØØ, 3ØØ, 3ØØ, 3ØØ, 935, 935, 49Ø
9ØØ BOXR%=21: BOXC%=23: BOXW%=35: BOXH%=3: GOSUB 66Ø
9Ø5 R%=22: C%=26: S$="     Enter desired PROG Number": GOSUB 6ØØ: GOSUB 995
91Ø R%=22: C%=26: S$="":   GOSUB 6ØØ:  GOSUB 5ØØ:  GOSUB 995
915 IF K%<1 OR K%>A% OR LEN(K$)>2 THEN 9ØØ
92Ø R%=22: C%=31: S$="Now Loading  "+PNF$(K%)+STRING$(2Ø,32):  GOSUB 6ØØ
925 GOSUB 66Ø: GOSUB 995: LOCATE 22,1,1
93Ø IF MSN%=2 THEN LOAD PNF$(K%)  ELSE  RUN PNF$(K%)
935 BOXR%=21: BOXC%=18: BOXW%=45: BOXH%=3: GOSUB 66Ø
94Ø R%=22: C%=21: S$="     Enter NUMBER of FileSpec to DELETE": GOSUB 6ØØ
945 GOSUB 995: R%=22: C%=21: S$="":   GOSUB 6ØØ:  GOSUB 5ØØ: GOSUB 995
95Ø IF K%<1 OR K%>A% OR LEN(K$)>2 THEN 89Ø
955 PNF$(K%)=LEFT$(PNM$,2)+PNF$(K%):  S$="Now DELETING  "+PNF$(K%)
96Ø R%=22: C%=26: S$=S$+STRING$(36-LEN(S$),32):  GOSUB 6ØØ:  KILL PNF$(K%)
965 R%=INT((K%-1)/5): C%=(K%-(R%*5)-1)*16+2: R%=R%+1Ø
97Ø S$=STRING$(15,176): GOSUB 6ØØ
975 R%=22: C%=21: S$="    DELETE another file ?   Y/N   ":    GOSUB 6ØØ
98Ø C%=POS(C):  GOSUB 5ØØ:  GOSUB 995
985 IF K$="Y" OR K$="y" THEN 935  ELSE GOSUB 99Ø: GOTO 3ØØ
99Ø FOR R%=23 TO 9 STEP -1: C%=1: S$=STRING$(8Ø,32): GOSUB 6ØØ: NEXT R%: RETURN
995 R%=2: C%=3: S$=DATE$: GOSUB 6ØØ: C%=66: S$=TIME$: GOSUB 6ØØ: RETURN
999 KEY OFF: LOCATE 22,1,1: END
```

PCM

ENTER YOUR DATA

ADD A BAR CODE READER AND SOFTWARE

A portable data collection system from Radio Shack puts all the power you need for fast and accurate inventory control— and it fits inside your briefcase.

Perfect for Retailers and Supervisors

From bar code reading to word processing, a Model 100 data collection system offers all the features retailers, auditors and warehouse supervisors need for accurate product inventory and reporting.

Our Bar Code Reader (26-1183, $99.95) makes the system faster and simpler to use than conventional inventorying methods, with their time-consuming product identification, counting and keyboard data entry.

Save Time with Scanner Technology

Just connect the pen-like optical scanner into the Model 100's standard

QUICKLY AND EASILY
TO RADIO SHACK'S PORTABLE MODEL 100

bar code reader interface, load the cassette software that's included (cassette recorder required) and you're ready to begin high-speed data entry from Plessey, Universal Product Code and 3 of 9 bar codes. Add our Additional Bar Code Reader software (26-3846, $19.95) and your portable system can read UPC-C, Codabar and Interleaved 2 of 5.

Create Your Own Inventory System

You can even create your own inventory system and bar codes with our Bar Code Generator software (26-3845, $34.95), and print bar code labels tailored to your specific business needs on any Radio Shack dot-matrix printer.

Computer Power to Finish the Job

Finish your inventory with a correction-free report written on your Model 100. You can even transmit the inventory information by telephone, using the Model 100's built-in phone modem. Data/Sort Plus software (26-3838, $49.95) lets you organize data files and update, merge and list information.

Find Out More Today

You can get a hands-on demonstration of our Model 100 and learn more about our portable data control system at Radio Shack Computer Centers and at participating Radio Shack stores and dealers.

Available at over 1200 Radio Shack Computer Centers and at participating Radio Shack stores and dealers.

Radio Shack
COMPUTER CENTERS
A DIVISION OF TANDY CORPORATION

Prices apply at Radio Shack Computer Centers and at participating Radio Shack stores and dealers

New 1986 Computer Catalog. Send me a copy.
Mail To: Radio Shack, Dept. 86-A-120, 300 One Tandy Center, Fort Worth, Texas 76102

NAME _____
COMPANY _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____
PHONE _____

Read disk sectors from BASIC

Create a Small Disk Zapper From Three Easy Subroutines

By Robert D. Covington

Have you ever accidentally typed FORMAT and entered in the wrong disk letter? Then, after seeing your error, stopped the format just after the first track was formatted?

How about the dreaded and unrecoverable "sector not found" error that seems to be on your original 8,000-part inventory disk as well as its backups?

What do you do about these problems? Crying always seems to make you feel a bit better but it usually never gets the data back. In the case of disk errors and deleted data, the only hope for recovery comes from a "disk zapper."

What is a disk zapper? It is a utility program that allows the entire disk to be accessed on a sector-by-sector basis. For example, if the boot track on a MS-DOS system were accidentally erased, it could be rebuilt by an experienced programmer with a disk zapper. In the case of a "sector not found" error, you could copy all the readable data on another disk and try to rebuild the data that is deleted (MS—DOS's RECOVER helps a lot in doing this).

In addition to the above items, a disk zapper can help recover erased files, rebuild destroyed directories, rebuild FATs (file allocation tables) and help you learn more about a disk's organization.

What does this have to do with subroutines, the topic of this column? In this month's "Subroutine City," I will be describing three subroutines that allow reading, writing and verifying sectors of data on a disk in BASIC. With these subroutines, a small disk zapper can easily be written in BASIC.

Disk Access Subroutines

When I orginally described the machine language MS-DOS access subroutine (August 1985), I stated that it was capable of not only accessing all of MS-DOS's system functions but also the actual BIOS. In this month's installment, I am going to use that ability to directly access the disk BIOS function INT 13H from BASIC. This will give BASIC the same disk access capability that MS-DOS has at a system level.

All three of the disk access subroutines in the program require four main parameters. The variable D holds the

---

(Bob Covington has been a computer programmer and consultant for the past six years, most recently focusing his attention on both the Model 100 and the 2000. He is also a technical writer and editor. Bob can be contacted at P.O. Box 37007, St. Louis, MO 63141.)

number of the disk drive to access. For example, to access drive B, D should contain a 1. The easiest way I have found to convert between the drive letter and its number is to subtract the ASCII value for the drive letter from 65 (see Line 580 of the program).

The variable T holds the track number to be accessed. Even though MS-DOS supports up to 1,024 tracks, the actual number used should be considerably less. On the Model 1000 and 1200, the floppy disk drives can access up to 40 tracks while the floppies in the Model 2000 support 80 tracks. In the case of a 10-megabyte hard disk drive, up to 306 tracks (sometimes called cylinders) can be accessed.

The variable H holds the head number or side of the disk to be accessed. Since a floppy disk has only two sides (I sure would like to see a three-sided floppy) obviously the number of heads is restricted to two (0 and 1). On a hard disk drive, however, multiple platters are used. In a standard 10-megabyte hard disk drive, two platters are used (that's four sides if I remember my 2-D geometry). In that case, the value in H ranges from zero to three.

Lastly, the variable S holds the sector number to be accessed. On a standard MS-DOS floppy format, nine 512-byte sectors are present on each side of a disk on a certain track. In some older IBM formats, only eight sectors per side per track are used.

In addition to the above entry variables, a 512-byte buffer needs to be set up for the source or destination of the disk operation. In Line 30 of the program, the array DT$( ) is initialized with 512 bytes broken up into 32 sections of 16 bytes. If you notice, the array is built from the last element to the first (backwards). This is done to make sure that BASIC allocates the strings sequentially in memory. If the strings were allocated from the first to the last element, BASIC might possibly scramble the variables throughout memory.

Since the data in the array DT$( ) must always be in sequential memory addresses, make sure to *never* modify the data in the string with a normal string function. If modifications must be done in the string, POKE the values into the array or use the MID$( )= statement (not function). This will keep BASIC from moving the string around during a string function evaluation.

Once all of the above entry requirements have been met, the subroutines at 17000, 18000 and 19000 respectively read, write and verify sectors of data on a disk. The sector read and write subroutines (17000 and 18000) use DT$( ) as a read and write buffer. The verify sector subroutine verifies the sector by attempting to read the sector into a dummy memory address. If the record can be read from the disk controller with no errors, the record is assumed to be correctly stored on disk. Since this entire operation is performed in the BIOS, no read/write buffer is required for this operation.

On return from the above subroutines, E contains the error status of the operation. The valid error statuses with their associated meanings are:

```
  0 — No error
  1 — Bad parameter or command
  2 — Address mark not found
  4 — Sector not found
  8 — DMA overrun
 16 — CRC error
 32 — FDC failure
 64 — Seek error
128 — Timeout error
```

Demonstration Program

Lines 40 through 999 of the program contain a simple disk sector viewer for demonstrating the read sector subroutine at Line 17000. This program displays the hexadecimal and ASCII representations of the data on any section of the disk. I have found this program to be useful for locating important information on the disk such as the boot sector, the directory sectors and data which was accidentally erased. The commands supported by this program are:

↑	Move back one record
↓	Move forward one record
SHIFT ↑	Move back one track
SHIFT ↓	Move forward one track
HOME	Move to start of disk
N	View other 256 bytes
H	Select new head
D	Select new drive
ESC	Exit to DOS

When executed, the program first asks for the drive to access. The drive letter entered here should not contain anything but the actual letter (no colon). Next, the computer displays the first 256 bytes on the disk (track 0, sector 1, head 0). At this point, the arrow keys can be used to scan through the tracks and sectors on the disk. Because of the size of the screen, the entire 512 bytes stored in a sector cannot be displayed on the screen at one time. To view the other 256 bytes of data, press the 'N' key.

If this program is executed on a standard Model 1000 or 1200, the variable MT needs to be changed to a 39 in Line 25. If hard drives are to be accessed, this value should be changed to a 305. By changing the value in MT, the maximum track number that the program will attempt to read is changed accordingly.

The program description table contains a complete line-by-line description of the program. This should help anyone who wants to extend the capabilities of this simple disk sector viewer.

Notes on the MS-DOS Access Subroutine

In the last three installments of "Subroutine City," the machine language MS-DOS access subroutine has been used extensively. When the subroutine was originally published (August 1985), I stated that it is imperative that the machine language subroutine does not move around in memory when BASIC cleans up the variable allocation. To avoid this problem, I executed a FRE( ) instruction in Line 10 to force BASIC to clean up the variables right after the machine language program is stored in memory. Since the string the machine language program is stored in is the first to be initialized, no other string should effect its allocation. Thus, the machine language program should stay in the same memory location no matter how many times BASIC cleans up the variable allocation.

All this logic is great but the "proof is in the pudding," so to speak. Unfortunately, I had the right idea but I did not quite follow through with it to the end. If you notice in Line 20 of the program, the FRE( ) instruction has a string parameter between the parenthesis. With a string (any string), the FRE( ) instruction will force BASIC to clean up the variable allocation and things will work as I stated above. If you look at last month's installment of "Subroutine City", however, you will notice that a numeric variable X is used. Unfortunately, with a numeric variable, the FRE( ) instruction does not force BASIC to do any reallocation, but only returns what memory is available at the current moment. Because of this, the subroutines in last month's articles will do all sorts of nasty things if BASIC ever decides to clean up the variable table (this usually happens in large programs

or programs that perform a lot of string operations). To keep this from happening, change the X=FRE(X) to X=FRE-(BIOS$) in Line 20.

If you look at the subroutine in the past three installments of "Subroutine City," you will notice that they are all designed to be incorporated into one large subroutine module. I took great care in making sure that all the subroutines that used the MS-DOS access subroutine would be compatible with each other. This allows all of the subroutines to be used together in one program.

Conclusion

With the subroutines presented in this article, a usable disk zapper can be easily constructed in BASIC. The demonstration program supplied at the beginning of the program basically has a "look but do not touch" attitude. Unless you are very familiar with the structure of an MS-DOS formatted disk and the disk operating system, I would suggest you do not add the ability to modify data on a disk to the program. If the wrong information is stored on a disk, the entire disk could be easily destroyed.

What else can these routines be used for other than a disk zapper? I have used them for converting various disk formats used on other machines to MS-DOS's format. This can be done by using BASIC to decode the directories of the foreign-formatted disk and then write the data to an MS-DOS formatted disk using BASIC's file I/O statements.

These routines can also be used for designing your very own disk operating system in BASIC. Granted, the DOS would be a bit slow, but wouldn't it be fun to make a UNIX-like operating system in BASIC? □

Description of Program

10	Preallocate all parameter variables and arrays.	90	Ask for drive number and display first sector on disk.	17000-17010	Read disk sector routine.
15	Set data pointer to machine language program data.	100	Main command loop. Get key from keyboard. Check key with valid command key list and execute the proper subroutine.	18000-18010	Write disk sector routine.
20	Load machine language program into BIOS$. Locate address of BIOS$ and store it in BIOS!	150-580	Command subroutines (documented with REMs).	19000-19020	Verify sector routine.
25	Define runtime constants.	800	Display record subroutine. Convert DT$( ) to a displayable form. P determines which 256 bytes of the 512-byte record are shown on the screen.	19020	Set up most machine language parameters from BASIC parameters, access MS-DOS subroutine and return with the error status.
30	Allocate a 512-byte buffer for sector data.			20000-20030	Set up machine language register parameters for INT 13H BIOS access.
40	Load legal command keys into KI$.			31000-50000	Documented with REMs (old subroutines).
50	Legal command key ASCII value data table.				

The listing:

```
1 ' *** MSDOS Function Subroutines for BASIC
2 ' *** By Robert D. Covington
10 DIM BIOS$,AX%,BX%,CX%,DX%,BP%,SI%,DI%,SW%,DT$(31)
15 RESTORE 50000
20 FOR X=1 TO 100:READ A:BIOS$=BIOS$+CHR$(A):NEXT:X=FRE(BIOS$):V=VARPTR(BIOS$):B
IOS!=PEEK(V+1)+PEEK(V+2)*256
25 SP$=SPACE$(255):NL$=CHR$(0):MT=79
30 FOR X=31 TO 0 STEP -1:DT$(X)=LEFT$(SP$,16):NEXT:X=FRE(DT$(0))
40 RESTORE 50:FOR X=1 TO 9:READ A:KI$=KI$+CHR$(A):NEXT
50 DATA 30,31,133,134,11,78,27,72,68
90 GOSUB 550
100 A$=INKEY$:IF A$="" THEN 100 ELSE ON INSTR(KI$,A$) GOSUB 150,200,250,300,350,
400,450,500,550:GOTO 100
150 ' *** Move back one record
160 S=S-1:IF S<1 THEN S=9:GOTO 250
170 GOTO 800
200 ' *** Move forward on record
210 S=S+1:IF S>9 THEN S=1:GOTO 300
220 GOTO 800
250 ' *** Move back one track
260 T=T-1:IF T<0 THEN T=MT
270 GOTO 800
300 ' *** Move forward one track
310 T=T+1:IF T>MT THEN T=0
320 GOTO 800
```

```
350 ' *** Move to start of disk
360 T=0:H=0:S=1:GOTO 800
400 ' *** View other 256 bytes in record
410 P=-(P=0):GOTO 900
450 ' *** Exit program
460 SYSTEM
500 ' *** Select other head
510 IF D<2 THEN H=-(H=0):GOTO 800
520 CLS
530 INPUT"Head number (0-7)";H:IF H<0 OR H>7 THEN PRINT"Illegal head number":GOT
O 530
540 GOTO 800
550 ' *** Select new drive
560 CLS
570 INPUT"View which drive (A-Z)";A$:IF A$<"A" OR A$>"Z" THEN PRINT"Illegal driv
e letter":GOTO 570
580 D=ASC(A$)-65:GOTO 350
800 ' *** Print new record
810 GOSUB 17000:IF E>0 THEN CLS:PRINT"Disk error!":END ELSE GOTO 900
900 ' *** Print binary data in DT$(X) on screen in hex/ascii form
910 CLS:AD=256*P:X3=P*16:FOR X=X3 TO X3+15:FOR X2=1 TO 16
920 A$=MID$(DT$(X),X2,1):HX$=HX$+RIGHT$("00"+HEX$(ASC(A$)),2)+" ":IF A$<" " THEN
 A$="."
930 AS$=AS$+A$:NEXT:PRINT RIGHT$("000"+HEX$(AD),3);"H";" => ";HX$;" <=> ";AS$:AS
$="":HX$="":AD=(X+1)*16:NEXT
940 PRINT:PRINT"Track:";T,"Sector:";S,"Head:";H,"View:";P:RETURN
999 END
```

```
17000 ' *** Read disk sector
17001 '  Entry:
17002 '          T, D, H, S, and DT$() same as routine at 20000
17003 '  Exit:
17004 '          E - Error status (0 - no error)
17005 '
17010 AX%=513:GOTO 19020
18000 ' *** Write disk sector
18001 '  Entry:
18002 '          T, D, H, S, and DT$() same as routine at 20000
18003 '  Exit:
18004 '          E - Error status (0 - no error)
18005 '
18010 AX%=769:GOTO 19020
19000 ' *** Verify sector
19001 '  Entry:
19002 '          T, D, H, S, and DT$() same as routine at 20000
19003 '  Exit:
19004 '          E - Error status (0 - no error)
19005 '
19010 AX%=1025
19020 GOSUB 20000:GOSUB 40000:E=INT(AX%/256):RETURN
20000 ' *** Setup variables for Floppy/Hard drive BIOS access
20001 '  Entry:
20002 '          D - Drive number (0-25 for A-Z)
20003 '          T - Track number (0-1023)
20004 '          H - Head number (0-7)
20005 '          S - Sector number (1-63)
20006 '  Exit:
20007 '          All disk description registers initialized
20010 V!=VARPTR(DT$(0)):A!=PEEK(V!+1)+PEEK(V!+2)*256:GOSUB 31000:BX%=A%:VV=A%
20015 I=19:IF D>1 THEN D=D OR 128
20020 A!=D+H*256:GOSUB 31000:DX%=A%
20025 XT=T/256:A!=((XT-INT(XT))*256)*256+(S OR (INT(XT)*64)):GOSUB 31000:CX%=A%
20030 RETURN
31000 ' *** Convert Integer to signed integer
31001 '  Entry:
31002 '          A! - Integer (0-65535)
31003 '  Exit:
31004 '          A% - Signed Integer (-32768 - 32767)
31005 '
31010 IF A!>32767 THEN A%=A!-65536! ELSE A%=A!
31015 RETURN
40000 ' *** Call DOS function
40001 ' I - Interrupt Number
40002 ' AX%, BX%, CX%, DX%, BP%, SI%, and DI% - Registers passed to and from DOS
40003 ' SW% - Status word
40004 '
40010 MID$(BIOS$,48,1)=CHR$(I):CALL BIOS!(AX%,BX%,CX%,DX%,BP%,SI%,DI%,SW%):RETUR
N
50000 ' Program:BIOS        Length: 100 bytes
50001 DATA 30,7,139,236,139,94,4,255,55,157,139,94,6,139,63,139,94,8,139,55,139,
94,12,139,23,139,94,14,139,15,139,94,18,139,7,139,94,16,139,31,139,110,10,139,11
0,0,205,33,85,83,139,236,139,94,8,156,143,7,139,94,10,137,63,139,94,12
50002 DATA 137,55,139,94,16,137,23,139,94,18,137,15,139,94,22,137,7,91,139,110,2
0,137,94,0,139,110,14,91,137,94,0,202,16,0
60000 A$=INKEY$:IF A$="" THEN 60000 ELSE PRINT ASC(A$):GOTO 60000
65000 SAVE"BDISK
```

PCM

The Gallery

By Wayne Sanders, Curator

This month's gallery exhibit comes to us from Harold Kemp of Gainsville, Fla.

The program, CIRCLE.BAS, is an interesting example of using simple BASIC commands such as CIRCLE to generate some spectacular effects. As the program moves along, the pattern and colors change.

Originally written for the Tandy 2000, we modified CIRCLE.BAS to run on both the Tandy 1000 and 2000. Simply change the first line in the program to suite your particular computer. Because of the nature of the program though, the effect is much more detailed and interesting when run on a 2000.

A cash prize of $50 is on its way to Harold. If you have something that you would like to submit to the Gallery, send it along! Please send the program and/or binary dump on disk along with some notes on how it was created. A color screen dump, while not necessary, would help. If your work is selected, you just might find yourself with an extra 50 bucks! □

The listing:

```
1000 MODEL=2000
1010 IF MODEL=2000 THEN SCREEN 3:XR=1:C1=4:C2=7:C3=3
1020 IF MODEL=1000 THEN CLEAR ,,,32768!:SCREEN 5:XR=.5:C1=4:C2=14:C3=15
1030 CLS:KEY ON:KEY OFF
1040 COLOR 1,1
1050 FOR E=1 TO 3 STEP 1
1060 FOR X=10*XR TO 100*XR STEP 2*XR
1070 CIRCLE (200*XR,100*XR),X,C1
1080 CIRCLE (400*XR,100*XR),X,C1
1090 CIRCLE (500*XR,200*XR),X,C1
1100 CIRCLE (100*XR,200*XR),X,C1
1110 CIRCLE (300*XR,200*XR),X,C1
1120 CIRCLE (300*XR,100*XR),X,C1
1130 CIRCLE (300*XR,300*XR),X,C1
1140 CIRCLE (200*XR,200*XR),X,C1
1150 CIRCLE (400*XR,200*XR),X,C1
1160 CIRCLE (400*XR,300*XR),X,C1
1170 CIRCLE (200*XR,300*XR),X,C1
1180 NEXT X
1190 FOR Y=10*XR TO 150*XR STEP 6*XR
1200 CIRCLE (300*XR,200*XR),Y,C2
1210 CIRCLE (300*XR,300*XR),Y,C2
1220 CIRCLE (300*XR,100*XR),Y,C2
1230 CIRCLE (200*XR,200*XR),Y,C2
1240 CIRCLE (400*XR,200*XR),Y,C2
1250 NEXT Y
1260 FOR Z=10*XR TO 50*XR STEP 18*XR
1270 CIRCLE (300*XR,100*XR),Z,C3
1280 CIRCLE (300*XR,200*XR),Z,C3
1290 CIRCLE (300*XR,300*XR),Z,C3
1300 CIRCLE (200*XR,200*XR),Z,C3
1310 CIRCLE (400*XR,200*XR),Z,C3
1320 NEXT Z
1330 NEXT E
1340 GOTO 1340
```

PCM


12
3
6
9

This program changes the Model 100's built-in clock into a grandfather/cuckoo/alarm clock and perpetual calendar of time keeping precision

It's About Time — Multi-Clock

By Aileen and John Cornman

The Model 100's built-in clock is a convenient feature since it can be accessed by BASIC programs. However, it uses 24-hour military time rather than standard 12-hour time, and it ignores leap years. This program makes the Model 100's time keeping ability more useful and adds an accurate calendar.

Multi-Clock displays the time in 12-hour format with numerals almost seven lines tall. *Multi-Clock* can function like a grandfather clock with chimes, a cuckoo clock complete with cuckoo bird and an alarm clock. It can also display a perpetual calendar a month at a time, starting with the current month. A help key is available to provide online instructions for using *Multi-Clock*'s features.

The Model 100 is a powerful computer you can take anywhere. Now you can take your cuckoo clock, grandfather clock, alarm clock and calendar with you in the same portable package.

Before running *Multi-Clock*, follow the instructions in the Model 100 owner's manual to set the internal clock to the correct time, day and date. Save files and programs on tape so you have at least 13,000 bytes of free memory.

Using the Program

When the program is run, the time, day, month and date are shown in oversized characters that can be read from a distance. The bottom line of the screen indicates the purpose of each eight function keys, F1 through F8. Pressing F7 presents a help menu from which you may request further information on any of the functions, or return to the clock display. Pressing F8 will end the *Multi-Clock* program and return to the Model 100's main menu.

Grandfather Chimes

To activate the grandfather chimes, press F5. The screen clears and when the clock display returns, the letters Chme on the bottom line appear in a black box to indicate the chimes are active. To deactivate the chimes, press F5 again and the box will disappear from around Chme.

When the chimes are on, *Multi-Clock* plays different musical chimes on the quarter, half and three-quarter hour. On the hour, the chimes are followed by a correlating number of gongs. If you wish to stop the chimes once they have started to play, press any letter key or the space bar (but this will leave the chimes active).

Cuckoo Clock

To activate the cuckoo, press F6. Ckoo appears in a black box when the cuckoo is active. In cuckoo mode, the cuckoo pops onto the screen and sings as much of his song as befits the quarter, half and three-quarter hour. On the whole hour, he follows his song with the number of cuckoos that correspond to the hour. He then retires to await his next performance. Figure 1 is an illustration of the clock display when the cuckoo has appeared.

To deactivate the cuckoo, press F6 again. If you wish to leave the cuckoo active, but do not wish to hear his whole song, press any letter key or the space bar to stop the song at that point.

Silent Clock

When neither the chimes nor cuckoo is active, *Multi-Clock* keeps the time silently.

Alarm Clock

The alarm may be set to any time as far as 23 hours and 59 minutes in advance of the current time. The four selectable alarm sounds include chimes, the cuckoo song, a slow beep and a fast beep.

Press F1 to pick the kind of sound you want to hear when the alarm goes off. After you make the selection from the alarm sound menu, the clock display will reappear. The alarm sound is independent of whether the cuckoo or chime modes are active.

Press F2 to set the time you want the alarm to go off. The set screen shows the alarm time that is currently set. You may simply press ENTER if that time is correct, or you may enter a new time in the same format as the one displayed. If you enter a new time, the set screen will be redisplayed to confirm the alarm setting. When you press ENTER as confirmation, the clock display will reappear and Set will be in a black box to indicate the alarm is set.

Press F3 to turn the alarm off. When the alarm is off, Set will no longer be in the black box.

To turn the alarm off while it is sounding, press any letter key or the space bar. If the alarm should happen to sound at a time when the cuckoo

---

(Aileen and John Cornman live in Battle Ground, Wash., where they own and operate their own computer business, Structured Software Services.)

or chimes are scheduled to play, the cuckoo or chimes will sound first, followed by the alarm.

Perpetual Calendar

Press F4 to view the calendar. The current month will be displayed with the current day highlighted as a point of reference (see Figure 2 for an illustration of the calendar display).

You may view the previous month by pressing the left arrow-key or the next month by pressing the right-arrow key. You may also go backward or forward as many months as desired using the arrow keys. When you press the 'C' key, it returns to the clock display.

The calendar accounts for leap years, and the program adjusts the Model 100's internal date to February 29 when appropriate.

Program Notes

Since comments are omitted from the program listing to conserve memory, the following notes are provided as a guide to the program's operation.

Line Numbers	Routine Function
10-22	Title and initialization
30-90	Main control loop after a function key interrupt
60-90	Normal main control loop
100-106	F1: Select alarm sound
110	Common exit point for function key routines
200-238	F2: Set alarm time
300	F3: Turn off alarm
400-407	F4: Calendar control loop
408-429	Display calendar of month CM, year CY
480-482	Print a day on calendar, highlight if today
490-491	Set ND to the number of days in the month in CM
492-499	Put days of week, outlines and prompts on the calendar screen. Wait for an arrow key or 'C' key
500-502	F5: Turn chimes on or off
600-602	F6: Turn cuckoo on or off
700-707	F7: Help menu display and control loop
710-714	Help screen for F1
720	Help screen for F2
730	Help screen for F3
740	Help screen for F4
750-752	Help screen for F5
760-762	Help screen for F6
770	Help screen for F7
780	Help screen for F8
790	Wait for a key; return with it in K$
800	F8: Return to Model 100 main menu
2000-2100	Play chime or cuckoo if on and the time is right
2200	Read a note from a DATA line and play it
3000-3406	Check alarm time; sound if on and time is right
4000-4072	Display and update clock face
4899-4909	Draw a seven-segment digit at X,Y; digit in 'N'
4910-4970	Seven routines each draw a digit segment
4980	Erase a digit from the screen
5010-5100	Ten routines each draw a different capital letter: A, D, F, J, M, N, O, S, T and W
5110	Draws left side vertical line for capital letters
5120	Draws right side vertical line for capital letters
5130-5292	Seventeen routines each draw a lowercase letter: a, b, c, d, e, g, h, i, l, n, o, p, r, y, u, v and y
5300-5310	Subroutines to draw segments of lowercase letters
10000-10010	Draw cuckoo bird on screen at X, Y
11000	Chime tune DATA line
12000	Cuckoo song DATA line

ROM Calls

17001	sets reverse character mode for printing white on black.
17006	sets normal character mode for printing black on white.

Significant Variables

- AL — Alarm sound. 1 = cuckoo song, 2 = chimes, 3 = slow beep, 4 = fast beep.
- CD — Calendar day. Starts off as the value of the current day of the month. Used by the calendar routine, along with CN, to determine the day of the week that the first day of the month to be displayed falls on.
- CF — Chime flag. When = 1, the chimes feature is turned on.
- CY — Calendar Year. Last two digits of the year to be displayed by the calendar routine.
- CM — Calendar month. Month to be displayed by calendar routine.
- CN — Calendar day number. Starts off as the number of the current day of the week (DN). Used by the calendar routine, along with CD, to determine the day of the week that the first day of the month to be displayed falls on.
- D1 — Tens' digit of the current date.
- D2 — Ones' digit of the current date.
- DA$ — Current day from Model 100 internal clock.
- DT$ — Current date from Model 100 internal clock.
- DN — Day number (1 through 7) of the current day of the week.
- DU — Duration of a note to be SOUNDed.
- H1 — Value of the tens' digit of the current hour.
- H2 — Value of the ones' digit of the current hour.
- HA — Old value of H1, used to tell when to change the tens' digit of the hour on the clock face.
- HB — Old value of H2, used to tell when to change the ones' digit of the hour on the clock face.
- HR — Value of the current hour.
- IR — Interrupt flag. When = 1 means clock display was interrupted by one of the function key routines, so clock needs to be redisplayed in its entirety.
- KF — Cuckoo flag. When = 1, the cuckoo feature is turned on.
- KT — The minute on which the

Figure 1: Multi-Clock's time and date display, showing the menu of options and the singing cuckoo bird.

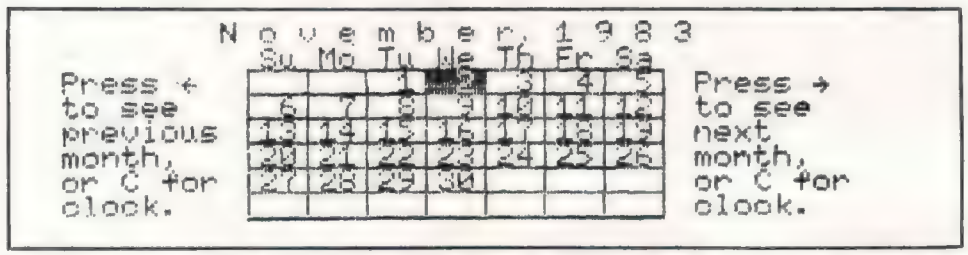


Figure 2: Multi-Clock's perpetual calendar. The current date is high-lighted by being printed in a black box.

cuckoo or chimes last sounded.

M1 — Value of the tens' digit of the current minute.

M2 — Value of the ones' digit of the current minute.

MA — Old value of M1, used to tell when to change the tens' digit of the minute on the clock face.

MB — Old value of M2, used to tell when to change the ones' digit of the minute on the clock face.

MF — Menu flag. 1 = need to redisplay menu line at bottom of the clock face; 0 = do not redisplay menu line.

MN — Value of the current minute.

MO — Value of the current month.

N — Value of the note to be SOUNDed. Also used to pass the digit to be displayed to the seven-segment digit drawing routine (as digit + 1).

N1 — Day of the week (1-7) of the first day of the month being displayed by the calendar routine.

N( ) — Array holding the values used in the SOUND statement to play the chimes and cuckoo song. N(1) through N(5) are note values, N(6) through N(9) are duration values. DATA Line 11000 is the chimes tune and DATA Line 12000 is the cuckoo tune. Each item of data consists of a tens' digit, which specifies the note value (1-5) and a ones' digit, which specifies the duration (6-9).

ND — Number of days in CM, the month displayed by the calendar.

NL — Day of the week (1-7) of the last day of the month being displayed by the calendar routine. Used to determine the day of the week the first day of the next month falls on if the user pages forward one month with the right arrow-key.

NT$ — Net time for alarm to go off, entered on set screen.

OM — Old value of SM on which the alarm last sounded. Used to keep the alarm from sounding again during the same minute if it is turned off, then immediately reset to the same time.

PA — Old value of PM indicator, used to tell when the AM/PM indicator should be changed on the clock face.

PM — PM flag. When = 1, current time is a PM time; 0 = AM.

SA$ — "AM" or "PM" to qualify the 12-hour time in ST$.

SF — Set flag. When = 1, the alarm is set to go off.

SH — Set hour. The hour for which the alarm is set.

SM — Set minute. The minute for which the alarm is set.

ST$ — The time for which the alarm is set, 12-hour format "hh:mm."

T — Coded note and duration value read from a data line.

TM$ — Current time from Model 100 internal clock.

TT — Number of notes to be played on the ¼, ½, ¾ or whole hour if the cuckoo or chimes are on.

X — X coordinate used to indicate where to draw the cuckoo or oversized letters and digits on the screen.

X1 — Old value of D1, used to tell when to change the tens' digit of the date on the clock face.

XA$ — Old value of DA$, used to tell when to change the day display on the clock face.

XO — Old value of MO, used to tell when to change the month on the clock face.

XT — Thickness of digits (number of dots thick minus one) to be displayed by the seven-segment digit drawing routine.

XW — Width of digits (number of dots wide minus one) to be displayed by the seven-segment digit drawing routine.

Y — Y coordinate used to indicate where to draw the cuckoo or oversized letters and digits on the screen. □

The listing:

```
10 '* CLOCK *
20 '* by Structured Software Services *
22 DEFINTA-Z:ST$="12:00":SA$="PM":SH=12:
SM=0:OM=99:KT=99:AL=3:N(1)=1567:N(2)=175
8:N(3)=1864:N(4)=2092:N(5)=2348:N(6)=16:
N(7)=32:N(8)=40:N(9)=80
30 CLS:PA=9:XA$="":HA=99:HB=99:MA=99:MB=
99:X1=99:XO=99:MF=1
40 KEYON:ONKEYGOSUB100,200,300,400,500,6
00,700,800
60 GOSUB4000
70 GOSUB2000
80 GOSUB3000
90 IFIR=1THENIR=0:GOTO30ELSE60
100 KEYOFF:CLS:PRINT:PRINT"(1) Cuckoo so
ng":PRINT"(2) Chimes":PRINT"(3) Slow bee
p":PRINT"(4) Fast beep"
102 PRINT:PRINT"Press number of desired
alarm ...";
104 K$=INKEY$:IFK$=""THEN104
106 AL=VAL(K$):IFAL<1ORAL>4THEN100
110 CLS:IR=1:FORI=1TO10:X$=INKEY$:NEXT:R
ETURN
200 KEYOFF:CLS:PRINT:PRINT"Alarm is now
set for ";ST$;" ";SA$
202 PRINT:PRINT"Enter new time, or just
press ENTER if":PRINT:LINEINPUT"setting
is correct: ";NT$
```

```
2Ø4 IFNT$=""THENSF=1:GOTO11Ø
2Ø6 C=INSTR(1,NT$,":"):IFC=ØTHEN2ØØ
2Ø8 NA$=RIGHT$(NT$,2):IFNA$="am"THENSA$=
"AM":GOTO214
21Ø IFNA$="pm"THENSA$="PM":GOTO214
212 IFNA$<>"AM"ANDNA$<>"PM"THEN2ØØELSESA
$=NA$
214 SH=VAL(LEFT$(NT$,C-1)):IFSH<1ORSH>12
THEN2ØØ
216 SM=VAL(MID$(NT$,C+1,2)):IFSM<ØORSM>5
9THEN2ØØ
218 ST$=""
22Ø ST$=ST$+RIGHT$(STR$(SH),LEN(STR$(SH)
)-1)+":"
222 IFSM<1ØTHENST$=ST$+"Ø"
224 ST$=ST$+RIGHT$(STR$(SM),LEN(STR$(SM)
)-1)
238 GOTO2ØØ
3ØØ KEYOFF:SF=Ø:GOTO11Ø
4ØØ IFMO=ØTHENRETURNELSEKEYOFF:CY=VAL(RI
GHT$(DT$,2)):CM=MO:CD=D1*1Ø+D2:CN=DN
4Ø1 CLS:GOSUB4Ø8
4Ø2 IFK$="C"ORK$="c"THEN11Ø
4Ø3 IFK$=CHR$(29)THEN4Ø6ELSECM=CM+1:IFCM
=13THENCM=1:CY=CY+1
4Ø5 CD=1:CN=NL+1:IFCN=8THENCN=1:GOTO4Ø1E
LSE4Ø1
4Ø6 CM=CM-1:IFCM=ØTHENCM=12:CY=CY-1
4Ø7 GOSUB49Ø:CD=ND:CN=N1-1:IFCN=ØTHENCN=
7:GOTO4Ø1ELSE4Ø1
4Ø8 CD=CDMOD7:IFCD=ØTHENCD=7
4Ø9 IFCD>CNTHENN1=CN+8-CDELSEN1=CN+1-CD
41Ø PRINT@7,"";:ONCMGOTO411,412,413,414,
415,416,417,418,419,42Ø,421,422
411 PRINT"  J a n u a r y";:GOTO423
412 PRINT" F e b r u a r y";:GOTO423
413 PRINT"    M a r c h";:GOTO423
414 PRINT"    A p r i l";:GOTO423
415 PRINT"      M a y";:GOTO423
416 PRINT"     J u n e";:GOTO423
417 PRINT"     J u l y";:GOTO423
418 PRINT"   A u g u s t";:GOTO423
419 PRINT"S e p t e m b e r";:GOTO423
42Ø PRINT"  O c t o b e r";:GOTO423
421 PRINT" N o v e m b e r";:GOTO423
422 PRINT" D e c e m b e r";
423 PRINT", 1 9";CY\1Ø;CHR$(8);CYMOD1Ø;S
PACE$(6)
424 GOSUB49Ø:I=1:P=9Ø:FORII=N1TO7:GOSUB4
8Ø:I=I+1:NEXTII
425 FORP=13ØTO21ØSTEP4Ø
426 FORII=1TO7:GOSUB48Ø:I=I+1:NEXTII:NEX
TP:NL=7
427 IFI>NDTHEN492
428 FORII=1TO7:IFI<=NDTHENGOSUB48Ø:NL=II
:I=I+1
429 NEXTII:P=29Ø:GOTO427
48Ø PP=P+(II-1)*3:IFCM=MOANDI=D1*1Ø+D2AN
```

```
DCY=VAL(RIGHT$(DT$,2))THENCALL17ØØ1:PX=(
PPMOD4Ø)*6:PY=(PP\4Ø)*8:LINE(PX-3,PY)-(P
X+15,PY+7),1,BF
482 PRINT@PP,USING"##";I;:CALL17ØØ6:RETU
RN
49Ø IFCM=4ORCM=6ORCM=9ORCM=11THENND=3Ø:R
ETURN
491 IFCM<>2THENND=31:RETURNELSEIFCYMOD4=
ØTHENND=29:RETURNELSEND=28:RETURN
492 PRINT@5Ø,"Su Mo Tu We Th Fr Sa"
493 FORI=57TO183STEP18:LINE(I,15)-(I,63)
:NEXT:FORI=15TO63STEP8:LINE(57,I)-(183,I
):NEXT
494 PRINT"Press ←";:PRINT@12Ø,"to see";:
PRINT@16Ø,"previous";:PRINT@2ØØ,"month,"
;:PRINT@24Ø,"or C for";:PRINT@28Ø,"clock
.";
495 PRINT@112,"Press →";:PRINT@152,"to s
ee";:PRINT@192,"next";:PRINT@232,"month,
";:PRINT@272,"or C for";:PRINT@312,"cloc
k.";
499 K$=INKEY$:IFK$="C"ORK$="c"ORK$=CHR$(
28)ORK$=CHR$(29)THENRETURNELSE499
5ØØ KEYOFF:IFCF=1THENCF=ØELSEKF=Ø:CF=1
5Ø2 GOTO11Ø
6ØØ KEYOFF:IFKF=1THENKF=ØELSECF=Ø:KF=1
6Ø2 GOTO11Ø
7ØØ KEYOFF:CLS:PRINT"Function keys do th
e following:"
7Ø1 PRINT"F1: Set alarm sound F5: Chimes
 on/off."
7Ø2 PRINT"F2: Set alarm time. F6: Cuckoo
 on/off."
7Ø3 PRINT"F3: Turn alarm off. F7: Get he
lp."
7Ø4 PRINT"F4: View calendar.  F8: Model
1ØØ Menu."
7Ø5 PRINT"For more help on a function, p
ress its  corresponding numeric key (not
 function key), or ENTER to return to cl
ock.";:LINE(Ø,7)-(239,7):LINE(Ø,39)-(239
,39)
7Ø6 GOSUB79Ø:IFK$=CHR$(13)THEN11ØELSEK=V
AL(K$)
7Ø7 IFK<1ORK>8THEN7Ø6ELSECLS:P$="Pressin
g F":ONKGOSUB71Ø,72Ø,73Ø,74Ø,75Ø,76Ø,77Ø
,78Ø:GOTO7ØØ
71Ø PRINT:PRINTP$;"1 allows you to pick
one of":PRINT:PRINT"the four kinds of so
unds the alarm can":PRINT:PRINT"make whe
n it goes off."
712 PRINT@28Ø,"(Press any key to return
to help menu.)";
714 GOSUB79Ø:RETURN
72Ø PRINT:PRINTP$;"2 sets the alarm to g
o off at the time you specify. On the cl
ock face,'Set' is printed in a black box
 when thealarm is set.":GOTO712
73Ø PRINT:PRINTP$;"3 turns the alarm off
```

```
. When   the alarm is off, 'Set' is not
in a     black box on the clock face. Wh
en the   alarm is sounding, pressing a l
etter or spacebar key will turn it off."
:GOTO712
74Ø PRINT:PRINTP$;"4 shows the calendar
for the  current month. You may page bac
kward andforward a month at a time. Toda
y's date appears in a black box.":GOTO71
2
75Ø PRINTP$;"5 turns the chimes on or of
f. On the clock face, 'Chme' is printed
in a black box when the chimes are on.
   If on, chimes will sound on the 1/4,"
752 PRINT"1/2, 3/4 and whole hour. Press
 a letter or spacebar key to stop the ch
ime.":GOTO712
76Ø PRINTP$;"6 turns the cuckoo on or of
f. On the clock face, 'Ckoo' is printed
in a black box when the cuckoo is on."
762 PRINT"If on, the cuckoo will come ou
t and singon the 1/4, 1/2, 3/4 and whole
 hour.    Press a letter or spacebar key
 to stop  the cuckoo's song.":GOTO712
77Ø PRINT:PRINTP$;"7 displays the help m
enu you":PRINT:PRINT"just saw.":GOTO712
78Ø PRINT:PRINTP$;"8 returns you to the
Model":PRINT:PRINT"1ØØ's main menu.":GOT
O712
79Ø K$=INKEY$:IFK$=""THEN79ØELSERETURN
8ØØ MENU
2ØØØ IF(KF=ØANDCF=Ø)ORKT=MNTHENRETURN
2ØØ2 IFMN=15ORMN=3ØORMN=45ORMN=ØTHENKEYO
FF:KT=MNELSERETURN
2ØØ4 IFKF=1THENX=146:Y=32:GOSUB1ØØØØ:RES
TORE12ØØØELSERESTORE11ØØØ
2Ø15 IFMN<>15THEN2Ø3Ø
2Ø16 IFKF=1THEN2Ø22
2Ø17 TT=4:GOSUB2Ø2Ø:GOTO21ØØ
2Ø2Ø FORI=1TOTT:GOSUB22ØØ:FORII=1TODU*2:
NEXTII:K$=INKEY$:IFK$<>""THENI=2Ø
2Ø21 NEXTI:RETURN
2Ø22 TT=4:GOSUB2Ø25:GOTO21ØØ
2Ø25 FORI=1TOTT:GOSUB22ØØ:K$=INKEY$:IFK$
<>""THENI=2Ø
2Ø26 NEXTI:RETURN
2Ø3Ø IFMN<>3ØTHEN2Ø45
2Ø31 IFKF=1THEN2Ø37
2Ø32 TT=7:GOSUB2Ø2Ø:GOTO21ØØ
2Ø37 TT=9:GOSUB2Ø25:GOTO21ØØ
2Ø45 IFMN<>45THEN2Ø6Ø
2Ø46 IFKF=1THEN2Ø53
2Ø47 TT=11:GOSUB2Ø2Ø:GOTO21ØØ
2Ø53 TT=13:GOSUB2Ø25:GOTO21ØØ
2Ø6Ø IFKF=1THEN2Ø8Ø
2Ø62 TT=15:GOSUB2Ø2Ø:IFK$<>""THEN21ØØ
2Ø64 FORI=1TOHR:SOUND4697,8Ø:FORII=1TO16
Ø:NEXTII
2Ø66 IFINKEY$<>""THENI=HR
2Ø68 NEXTI:GOTO21ØØ
2Ø8Ø TT=19:GOSUB2Ø25:IFK$<>""THEN21ØØ
2Ø82 FORI=1TOHR:SOUND1567,16:SOUND1864,3
2:FORII=1TO16Ø:NEXTII
2Ø84 IFINKEY$<>""THENI=HR
2Ø86 NEXTI
21ØØ LINE(146,27)-(171,52),Ø,BF:FORI=1TO
1Ø:K$=INKEY$:NEXT:KEYON:RETURN
22ØØ READT:DU=N(TMOD1Ø):N=N(T\1Ø):IFN>ØT
HENSOUNDN,DU:RETURNELSEFORII=1TODU*1Ø:NE
XTII:RETURN
3ØØØ IFSF=ØORSM=OMTHENRETURN
3ØØ2 IF(SH<>HR)OR(SM<>MN)OR(SA$="AM"ANDP
```

```
M=1)OR(SA$="PM"ANDPM=Ø)THENRETURNELSEKEY
OFF
3ØØ4 IFSF=ØTHENOM=SM:GOTO11Ø
3ØØ6 ONALGOTO31ØØ,32ØØ,33ØØ,34ØØ
31ØØ RESTORE12ØØØ:TT=19:GOSUB2Ø25:GOTO34
Ø4
32ØØ RESTORE11ØØØ:TT=15:GOSUB2Ø2Ø:GOTO34
Ø4
33ØØ BEEP:FORI=1TO16ØØ:NEXT:GOTO34Ø2
34ØØ BEEP:FORI=1TO2ØØ:NEXT
34Ø2 K$=INKEY$
34Ø4 IFK$<>""THENSF=Ø
34Ø6 GOTO3ØØ4
4ØØØ TM$=TIME$:DT$=DATE$:DA$=DAY$:IFDT$<
>DATE$ORDA$<>DAY$THEN4ØØØ
4ØØ1 HR=VAL(LEFT$(TM$,2)):MN=VAL(MID$(TM
$,4,2)):IFHR=12THENPM=1:GOTO4ØØ3
4ØØ2 IFHR>12THENHR=HR-12:PM=1ELSEPM=Ø:IF
HR=ØTHENHR=12
4ØØ3 H1=HR\1Ø:H2=HRMOD1Ø:M1=MN\1Ø:M2=MNM
OD1Ø:XT=4:XW=25:Y=2
4ØØ4 IFH1<>HATHENIFH1=1THENLINE(Ø,2)-(4,
52),1,BFELSELINE(Ø,2)-(4,52),Ø,BF
4ØØ8 HA=H1:IFH2<>HBTHENX=14:N=H2+1:GOSUB
4899:LINE(47,15)-(51,19),1,BF:LINE(47,35
)-(51,39),1,BF:HB=H2
4Ø12 IFM1<>MATHENX=59:N=M1+1:GOSUB4899:M
A=M1
4Ø14 IFM2<>MBTHENX=94:N=M2+1:GOSUB4899:I
FMB<>99THENOM=99:MB=M2ELSEMB=M2
4Ø16 IFPM=PATHEN4Ø2Ø
4Ø18 IFPM=1THENCALL17ØØ1:PRINT@221,"PM";
:CALL17ØØ6:LINE(125,39)-(137,47),1,BELSE
LINE(125,39)-(137,47),Ø,BF
4Ø2Ø PA=PM:IFDA$=XA$THEN4Ø48
4Ø21 IFLEFT$(DT$,5)="Ø3/Ø1"ANDXO=2ANDVAL
(RIGHT$(DT$,2))MOD4=ØANDD1*1Ø+D2=28THEND
ATE$="Ø2/29/"+RIGHT$(DT$,2):GOTO4ØØØ
4Ø22 LINE(126,2)-(177,25),Ø,BF:X=126:Y=2
4Ø24 IFDA$="Mon"THENDN=2:GOSUB5Ø5Ø:X=144
:GOSUB523Ø:X=162:GOSUB522Ø:GOTO4Ø38
4Ø26 IFDA$="Tue"THENDN=3:GOSUB5Ø9Ø:X=144
:GOSUB527Ø:X=162:GOSUB517Ø:GOTO4Ø38
4Ø28 IFDA$="Wed"THENDN=4:GOSUB51ØØ:X=144
:GOSUB517Ø:X=162:GOSUB516Ø:GOTO4Ø38
4Ø3Ø IFDA$="Thu"THENDN=5:GOSUB5Ø9Ø:X=144
:GOSUB519Ø:X=162:GOSUB527Ø:GOTO4Ø38
4Ø32 IFDA$="Fri"THENDN=6:GOSUB5Ø3Ø:X=144
:GOSUB525Ø:X=162:GOSUB52ØØ:GOTO4Ø38
4Ø34 GOSUB5Ø8Ø:X=144:IFDA$="Sat"THENDN=7
:GOSUB513Ø:X=162:GOSUB526Ø:GOTO4Ø38
4Ø36 DN=1:GOSUB527Ø:X=162:GOSUB522Ø
4Ø38 XA$=DA$:MO=VAL(LEFT$(DT$,2)):IFMO=X
OTHEN4Ø42ELSEX=186:LINE(X,Y)-(236,25),Ø,
BF
4Ø4Ø ONMOGOSUB4Ø61,4Ø62,4Ø63,4Ø64,4Ø65,4
```

```
Ø66,4Ø67,4Ø68,4Ø69,4Ø7Ø,4Ø71,4Ø72
4Ø42 XO=MO:D1=VAL(MID$(DT$,4,1)):D2=VAL(
MID$(DT$,5,1))
4Ø44 XW=11:XT=3:Y=3Ø:IFD1<>X1THENX=199:L
INE(X,Y)-(21Ø,52),Ø,BF:IFD1<>ØTHENN=D1+1
:GOSUB4899
4Ø46 X=215:LINE(X,Y)-(226,52),Ø,BF:N=D2+
1:GOSUB4899:X1=D1
4Ø48 IFMF=ØTHENRETURNELSEPRINT@28Ø,"Alrm
 ";:IFSF=ØTHENPRINT"Set  ";:GOTO4Ø5Ø
4Ø49 CALL17ØØ1:PRINT"Set";:CALL17ØØ6:PRI
NT"  ";:LINE(29,55)-(47,63),1,B
4Ø5Ø PRINT"Off  Cldr ";:IFCF=ØTHENPRINT"
Chme ";:GOTO4Ø52
4Ø51 CALL17ØØ1:PRINT"Chme";:CALL17ØØ6:PR
INT" ";:LINE(119,55)-(143,63),1,B
4Ø52 IFKF=ØTHENPRINT"Ckoo ";:GOTO4Ø54
4Ø53 CALL17ØØ1:PRINT"Ckoo";:CALL17ØØ6:PR
INT" ";:LINE(149,55)-(173,63),1,B
4Ø54 PRINT"Help Menu";:MF=Ø
4Ø6Ø RETURN
4Ø61 GOSUB5Ø4Ø:X=2Ø4:GOSUB513Ø:X=222:GOS
UB522Ø:RETURN
4Ø62 GOSUB5Ø3Ø:X=2Ø4:GOSUB517Ø:X=222:GOS
UB514Ø:RETURN
4Ø63 GOSUB5Ø5Ø:X=2Ø4:GOSUB513Ø:X=222:GOS
UB525Ø:RETURN
4Ø64 GOSUB5Ø1Ø:X=2Ø4:GOSUB524Ø:X=222:GOS
UB525Ø:RETURN
4Ø65 GOSUB5Ø5Ø:X=2Ø4:GOSUB513Ø:X=222:GOS
UB529Ø:RETURN
4Ø66 GOSUB5Ø4Ø:X=2Ø4:GOSUB527Ø:X=222:GOS
UB522Ø:RETURN
4Ø67 GOSUB5Ø4Ø:X=2Ø4:GOSUB521Ø:X=222:GOS
UB529Ø:RETURN
4Ø68 GOSUB5Ø1Ø:X=2Ø4:GOSUB527Ø:X=222:GOS
UB518Ø:RETURN
4Ø69 GOSUB5Ø8Ø:X=2Ø4:GOSUB517Ø:X=222:GOS
UB524Ø:RETURN
4Ø7Ø GOSUB5Ø7Ø:X=2Ø4:GOSUB515Ø:X=222:GOS
UB526Ø:RETURN
4Ø71 GOSUB5Ø6Ø:X=2Ø4:GOSUB523Ø:X=222:GOS
UB528Ø:RETURN
4Ø72 GOSUB5Ø2Ø:X=2Ø4:GOSUB517Ø:X=222:GOS
UB515Ø:RETURN
4899 ONNGOTO49ØØ,49Ø1,49Ø2,49Ø3,49Ø4,49Ø
5,49Ø6,49Ø7,49Ø8,49Ø9
49ØØ GOSUB498Ø:GOSUB491Ø:GOSUB492Ø:GOSUB
493Ø:GOSUB494Ø:GOSUB495Ø:GOSUB496Ø:RETUR
N
49Ø1 GOSUB498Ø:GOSUB492Ø:GOSUB493Ø:RETUR
N
49Ø2 GOSUB498Ø:GOSUB491Ø:GOSUB492Ø:GOSUB
494Ø:GOSUB495Ø:GOSUB497Ø:RETURN
49Ø3 GOSUB498Ø:GOSUB491Ø:GOSUB492Ø:GOSUB
493Ø:GOSUB494Ø:GOSUB497Ø:RETURN
49Ø4 GOSUB498Ø:GOSUB492Ø:GOSUB493Ø:GOSUB
496Ø:GOSUB497Ø:RETURN
49Ø5 GOSUB498Ø:GOSUB491Ø:GOSUB493Ø:GOSUB
494Ø:GOSUB496Ø:GOSUB497Ø:RETURN
49Ø6 GOSUB498Ø:GOSUB493Ø:GOSUB494Ø:GOSUB
495Ø:GOSUB496Ø:GOSUB497Ø:RETURN
49Ø7 GOSUB498Ø:GOSUB491Ø:GOSUB492Ø:GOSUB
493Ø:RETURN
49Ø8 GOSUB49ØØ:GOSUB497Ø:RETURN
49Ø9 GOSUB49Ø4:GOSUB491Ø:RETURN
491Ø LINE(X,Y)-(X+XW,Y+XT),1,BF:RETURN
492Ø LINE(X+XW,Y)-(X+XW-XT,Y+XW),1,BF:RE
TURN
493Ø LINE(X+XW-XT,Y+XW)-(X+XW,Y+2*XW),1,
BF:RETURN
494Ø LINE(X,Y+2*XW)-(X+XW,Y+2*XW-XT),1,B
F:RETURN
495Ø LINE(X,Y+XW)-(X+XT,Y+2*XW),1,BF:RET
URN
496Ø LINE(X,Y)-(X+XT,Y+XW),1,BF:RETURN
497Ø LINE(X,Y+XW-XT\2)-(X+XW,Y+XW-XT\2+X
T),1,BF:RETURN
498Ø LINE(X,Y)-(X+XW,Y+2*XW),Ø,BF:RETURN
5Ø1Ø LINE(X,Y+6)-(X+2,Y+2Ø),1,BF:LINE(X+
12,Y+6)-(X+14,Y+2Ø),1,BF:LINE(X+3,Y+12)-
(X+11,Y+14),1,BF:LINE(X+6,Y)-(X+8,Y+3),1
,BF
5Ø12 FORI=Y+1TOY+4:LINE(X+5,I)-(X+1,I+4)
:LINE(X+9,I)-(X+13,I+4):NEXT:RETURN
5Ø2Ø LINE(X,Y)-(X+9,Y+2),1,BF:LINE(X,Y+1
8)-(X+9,Y+2Ø),1,BF:LINE(X+3,Y+3)-(X+5,Y+
17),1,BF:LINE(X+12,Y+4)-(X+14,Y+16),1,BF
5Ø22 FORI=XTOX+3:LINE(I+8,Y)-(I+11,Y+3):
LINE(I+8,Y+2Ø)-(I+11,Y+17):NEXT:RETURN
5Ø3Ø GOSUB511Ø:LINE(X+3,Y)-(X+14,Y+2),1,
BF:LINE(X+3,Y+9)-(X+11,Y+11),1,BF:RETURN
5Ø4Ø LINE(X+6,Y)-(X+14,Y+2),1,BF:LINE(X+
9,Y+3)-(X+11,Y+16),1,BF:LINE(X,Y+12)-(X+
2,Y+16),1,BF:LINE(X+5,Y+18)-(X+6,Y+18)
5Ø42 FORI=XTOX+3:LINE(I,Y+17)-(I+3,Y+2Ø)
:LINE(I+5,Y+2Ø)-(I+8,Y+17):NEXT:RETURN
5Ø5Ø GOSUB511Ø:GOSUB512Ø:FORI=Y+1TOY+4:L
INE(X+3,I)-(X+7,I+4):LINE(X+11,I)-(X+7,I
+4):NEXT:RETURN
5Ø6Ø GOSUB511Ø:GOSUB512Ø:FORI=YTOY+3:LIN
E(X+2,I)-(X+12,I+17):NEXT:RETURN
5Ø7Ø LINE(X,Y+4)-(X+2,Y+16),1,BF:LINE(X+
12,Y+4)-(X+14,Y+16),1,BF
5Ø71 LINE(X+5,Y)-(X+9,Y+2),1,BF:LINE(X+5
,Y+18)-(X+9,Y+2Ø),1,BF
5Ø72 FORI=XTOX+3:LINE(I,Y+3)-(I+3,Y):LIN
E(I,Y+17)-(I+3,Y+2Ø):NEXT:GOTO5Ø22
5Ø8Ø LINE(X,Y+4)-(X+2,Y+7),1,BF:LINE(X+1
2,Y+4)-(X+14,Y+5),1,BF:LINE(X+5,Y+9)-(X+
9,Y+11),1,BF:LINE(X+12,Y+13)-(X+14,Y+16)
,1,BF:LINE(X,Y+15)-(X+2,Y+16),1,BF
5Ø82 FORI=XTOX+3:LINE(I,Y+8)-(I+3,Y+11):
LINE(I+8,Y+9)-(I+11,Y+12):NEXT:GOTO5Ø71
5Ø9Ø LINE(X,Y)-(X+14,Y+2),1,BF:LINE(X+6,
Y+3)-(X+8,Y+2Ø),1,BF:RETURN
51ØØ GOSUB511Ø:GOSUB512Ø:FORI=Y+16TOY+19
:LINE(X+3,I)-(X+7,I-4):LINE(X+11,I)-(X+7
```

```
,I-4):NEXT:RETURN
511Ø LINE(X,Y)-(X+2,Y+2Ø),1,BF:RETURN
512Ø LINE(X+12,Y)-(X+14,Y+2Ø),1,BF:RETUR
N
513Ø GOSUB53ØØ:GOSUB53Ø1:GOSUB53Ø2:GOSUB
53Ø4:GOSUB53Ø9:LINE(X,Y+15)-(X+2,Y+17),1
,BF:GOSUB53Ø7:GOSUB53Ø6
5132 YY=Y:Y=Y+6:GOSUB53Ø5:Y=YY:RETURN
514Ø GOSUB511Ø:GOSUB53Ø4:GOSUB53Ø6:GOSUB
53Ø8:LINE(X+6,Y+6)-(X+11,Y+8),1,BF:LINE(
X+6,Y+18)-(X+11,Y+2Ø),1,BF
5142 XX=X:X=X+3:GOSUB53Ø5:GOSUB53Ø7:X=XX
:YY=Y:Y=Y+3:GOSUB53Ø5:Y=YY-3:GOSUB53Ø7:Y
=YY:RETURN
515Ø GOSUB523Ø:LINE(X+12,Y+12)-(X+14,Y+1
4),Ø,BF:RETURN
516Ø GOSUB512Ø:GOSUB53Ø3:GOSUB53Ø5:GOSUB
53Ø7:LINE(X+3,Y+6)-(X+8,Y+8),1,BF:LINE(X
+3,Y+18)-(X+8,Y+2Ø),1,BF
5162 XX=X:X=X-3:GOSUB53Ø6:GOSUB53Ø8:X=XX
:YY=Y:Y=Y+3:GOSUB53Ø6:Y=YY-3:GOSUB53Ø8:Y
=YY:RETURN
517Ø GOSUB53ØØ:GOSUB53Ø1:GOSUB53Ø2:GOSUB
53Ø3:LINE(X+12,Y+9)-(X+14,Y+14),1,BF:GOS
UB53Ø6:GOSUB53Ø5:GOSUB53Ø7:RETURN
518Ø GOSUB531Ø
5182 LINE(X+3,Y+6)-(X+8,Y+8),1,BF:LINE(X
+3,Y+15)-(X+8,Y+17),1,BF:LINE(X,Y+9)-(X+
2,Y+14),1,BF:LINE(X+9,Y+9)-(X+11,Y+14),1
,BF
5184 GOSUB53Ø5:YY=Y:Y=Y-3:GOSUB53Ø7:XX=X
:X=X-3:GOSUB53Ø8:Y=YY:GOSUB53Ø6:X=XX:RET
URN
519Ø GOSUB511Ø:GOSUB53ØØ:GOSUB53Ø4:GOSUB
53Ø9:GOSUB53Ø6:RETURN
52ØØ GOSUB53Ø2:LINE(X+6,Y+6)-(X+8,Y+17),
1,BF:LINE(X+3,Y+6)-(X+5,Y+8),1,BF:LINE(X
+6,Y)-(X+8,Y+2),1,BF:RETURN
521Ø GOSUB53Ø2:LINE(X+6,Y)-(X+8,Y+17),1,
BF:LINE(X+3,Y)-(X+5,Y+2),1,BF:RETURN
522Ø GOSUB525Ø:GOSUB53Ø4:GOSUB53Ø9:RETUR
N
523Ø GOSUB53ØØ:GOSUB53Ø2:GOSUB53Ø3:GOSUB
53Ø4:GOSUB53Ø5:GOSUB53Ø6:GOSUB53Ø7:GOSUB
53Ø8:RETURN
524Ø LINE(X,Y+6)-(X+2,Y+23),1,BF:XX=X:X=
X+3:GOSUB5182:X=XX:RETURN
525Ø LINE(X,Y+6)-(X+2,Y+2Ø),1,BF:LINE(X+
6,Y+6)-(X+11,Y+8),1,BF:GOSUB53Ø6:XX=X:X=
X+3:GOSUB53Ø5:X=XX:YY=Y:Y=Y+3:GOSUB53Ø5:
Y=YY:RETURN
526Ø LINE(X+3,Y)-(X+5,Y+17),1,BF:LINE(X,
Y+6)-(X+11,Y+8),1,BF:LINE(X+6,Y+18)-(X+1
1,Y+2Ø),1,BF:GOSUB53Ø8:XX=X:X=X+3:GOSUB5
3Ø7:X=XX:RETURN
527Ø LINE(X,Y+6)-(X+2,Y+17),1,BF:LINE(X+
9,Y+6)-(X+11,Y+17),1,BF:LINE(X+3,Y+18)-(
X+8,Y+2Ø),1,BF
5272 GOSUB53Ø7:XX=X:X=X+9:GOSUB53Ø7:X=XX
-3:GOSUB53Ø8:X=XX:RETURN
528Ø FORI=XTOX+2:LINE(I,Y+6)-(I+6,Y+2Ø):
LINE(I+12,Y+6)-(I+6,Y+2Ø):NEXT:RETURN
529Ø LINE(X,Y+6)-(X+2,Y+14),1,BF:LINE(X+
3,Y+15)-(X+8,Y+17),1,BF:GOSUB531Ø
5292 YY=Y:Y=Y-3:GOSUB53Ø7:XX=X:X=X-3:GOS
UB53Ø8:X=XX:Y=Y-3:GOSUB53Ø8:Y=YY:RETURN
53ØØ LINE(X+3,Y+6)-(X+11,Y+8),1,BF:RETUR
N
53Ø1 LINE(X+3,Y+12)-(X+11,Y+14),1,BF:RET
URN
53Ø2 LINE(X+3,Y+18)-(X+11,Y+2Ø),1,BF:RET
URN
53Ø3 LINE(X,Y+9)-(X+2,Y+17),1,BF:RETURN
53Ø4 LINE(X+12,Y+9)-(X+14,Y+17),1,BF:RET
URN
53Ø5 FORI=XTOX+3:LINE(I,Y+9)-(I+3,Y+6):N
EXT:RETURN
53Ø6 FORI=X+8TOX+11:LINE(I,Y+6)-(I+3,Y+9
):NEXT:RETURN
53Ø7 FORI=XTOX+3:LINE(I,Y+17)-(I+3,Y+2Ø)
:NEXT:RETURN
53Ø8 FORI=X+8TOX+11:LINE(I,Y+2Ø)-(I+3,Y+
17):NEXT:RETURN
53Ø9 LINE(X+12,Y+18)-(X+14,Y+2Ø),1,BF:RE
TURN
531Ø LINE(X+12,Y+6)-(X+14,Y+2Ø),1,BF:LIN
E(X+3,Y+21)-(X+11,Y+23),1,BF:YY=Y:Y=Y+3:
GOSUB53Ø8:Y=YY:RETURN
1ØØØØ LINE(X,Y)-(X+5,Y-5):LINE(X+1,Y)-(X
+4,Y):PSET(X+5,Y-3):PSET(X+6,Y-5):LINE(X
+7,Y-5)-(X+9,Y-3):LINE(X+9,Y-2)-(X+9,Y-1
)
1ØØØ2 LINE(X+8,Y)-(X+1Ø,Y):LINE(X+5,Y+1)
-(X+7,Y+3),1,BF:PRESET(X+5,Y+2):PRESET(X
+7,Y+3):LINE(X+11,Y+1)-(X+19,Y+9):LINE(X
+12,Y+1)-(X+18,Y+7)
1ØØØ4 LINE(X+5,Y+4)-(X+8,Y+11):PSET(X+7,
Y+7):PSET(X+8,Y+4):LINE(X+8,Y+5)-(X+13,Y
+1Ø):LINE(X+14,Y+1Ø)-(X+17,Y+12):LINE(X+
17,Y+8)-(X+18,Y+9)
1ØØØ6 LINE(X+18,Y+1Ø)-(X+19,Y+12),1,BF:L
INE(X+9,Y+11)-(X+13,Y+15):LINE(X+1Ø,Y+11
)-(X+11,Y+12):LINE(X+11,Y+14)-(X+12,Y+15
):PSET(X+19,Y+13)
1ØØØ8 LINE(X+2Ø,Y+13)-(X+23,Y+16):LINE(X
+19,Y+14)-(X+21,Y+16):LINE(X+19,Y+16)-(X
+21,Y+18):LINE(X+22,Y+17)-(X+24,Y+19),1,
BF:PRESET(X+22,Y+19)
1ØØ1Ø LINE(X+24,Y+19)-(X+25,Y+2Ø),1,BF:L
INE(X+16,Y+15)-(X+18,Y+15):LINE(X+13,Y+1
5)-(X+15,Y+19),1,B:LINE(X+1Ø,Y+19)-(X+12
,Y+19):RETURN
11ØØØ DATA48,18,38,58,48,18,39,48,18,38,
58,48,18,39,8
12ØØØ DATA16,37,16,37,16,26,36,46,37,16,
37,16,37,16,26,36,46,57,7
```

PCM

Mailing labels from a spreadsheet? Richard White shows you how!

Marrying BASIC and a Spreadsheet

By Richard A. White
PCM Contributing Editor

Over the past few months, Danny Humphress has been showing you how to program *dBASE II* to handle a mailing list and print mailing labels. Use of a database manager utility is but one way to skin the mailing label cat. A mailing list can also be kept in a spreadsheet.

I have a nearly 400-name list of vendor representatives in *Lotus 1-2-3*. This was very easy to set up. List maintenance is also quick and easy. But, there is no way I can print mailing labels from *Lotus*. Does this mean that you need to go out and buy a database manager to go with your spreadsheet to do your mailing lists? Not necessarily. This month we will give you a simple BASIC program that will take an ASCII file produced by your spreadsheet and print mailing labels from it.

The idea of writing programs to do special manipulations of data files is as old as computers. In recent years, specialized software to do many of these tasks has reduced the amount of customizing that the user needs to do. Still, if you are going to try to stay with generalized utility programs, you are going to need to do some programming. It may be done using a language provided by the utility itself as in the *dBASE II* case. If you know the structure of the database, the programming can be done in BASIC or any other language you might choose.

If your spreadsheet can produce an ASCII disk file, you can make a file of known structure. When you choose the printer function in *VisiCalc* or *Lotus 1-2-3* with /P, you are given a choice of "printer" or "file." If you choose the printer, a stream of number codes for the letters to be printed is sent to the printer. If you choose "file," the same stream is sent to a file on disk which will be the famous ASCII file, subject of Bill Barden's ode to sequential files in the August 1985 issue of PCM.

Now that the file is there, we need to discuss its structure. Bill was writing about finding particular characters or character strings in the file and converting these to something else. He did not need to worry about where these strings were in the file or whether there were any specific arrangements for the data. Our problem is somewhat different since to print a mailing label, we need to find a name and print it on one line, find an associated address and print it on the next line and so on.

A spreadsheet makes a simple but effective file manager. To understand file structure, we need to discuss some basic concepts of file management. A file is a collection of records. This might bring to mind phonograph records which you might have quite a few of. You could make a file of the titles of all the albums. Widen Column A to 20 or 25 characters and type the names into the cells down the column.

OK, they are out of any logical order so there needs to be some sorting. Spreadsheets sort by exchanging columns or rows. Everything in the row or column is swapped with the next. This is not a mindless swapping. The spreadsheet looks within a range you give it and does the sorting according to the contents of the cells in the range. At this point, to sort the album file, you would use a range in Column A to include all the titles. The sort function works from the left most character in each name comparing characters with the next name until a mismatch is found. A swap occurs according to fixed rules.

Sorting can arrange the records in ascending order where the smallest is first or in descending order where the smallest is last. Okay for numbers, but which is smaller, A or B?

This brings us right back to the ASCII code numbers for each character. All the characters you can type from the keyboard and then some have a number assigned. That number is the ASCII value and it's that number that the computer uses to decide if A is larger than B. A's ASCII value is 66 while B's is 67, so the computer thinks B is bigger and you aren't going to change its mind.

Let's check out the rats in ASCII's woodpile. Here is a file of names of computer books on my shelf sorted in ascending order.

```
 PROGRAMMING THE Z80
1-2-3 A TO Z
BASIC09 TOURGUIDE
INTRODUCTION TO PASCAL
THE C PROGRAMMING LANGUAGE
cOMPUTERS FOR TECHINCIANS
```

A record with a leading space always sorts to the top of an ascending sort. You may not notice the space and wonder how a record got so out of order. That's how PROGRAMMING THE Z80 got to the top.

Numbers have lower ASCII values than letters, so 1-2-3 A to Z is just where it should be. If you enter dates like 7/4/85 and 10/12/85, Month 10

(Richard White has a long background with microcomputers and specializes in BASIC programming. With Don Dollberg, he is the author of the TIMS *database management program for the Color Computer.)*

will sort before Month 7. This is one reason you see computer produced dates like 07/04/85. The leading 0 sorts ahead of 1 and puts July before October.

Finally, lowercase letters sort after all uppercase letters. There are some punctuation and special characters before the numerals and some after, but all sort before the alphabet.

The album file only contains the names of the albums. These are only in Column A. Each row is a record and each has only one field in use, the Column A cell's. Lets call column A the "name" field.

There is other data that is associated with each album such as performers, publisher, type of music, date of publication, your rating of the performance and playing time to name a few. A column could be devoted to each of these data types with the particular data entered in the proper column in the row with the title to which it pertains. Again, each row is a record which now would have a number of fields of data. You have done this sort of list on paper before. But, you probably were not thinking in terms of records and fields when you did.

Now your sorting options become broader and you can do a number of sorts on the file. First, let's sort the whole file by type of music. You call the sort again and give for the range the column containing the music type data and include all the rows in use.

With that sort made, you might want the hard rock type sorted by performer. All hard rock albums should now be in one group in the spreadsheet. Sort using the performer column, but limit the range to only those rows that include the hard rock entries. You could as well have sorted hard rock by title or even year of publication.

Which way should the file be stored on disk? It's something to think about. If you have spent some time making multiple sorts, think about storing the file that way. You can always load it and re-sort it if you want it arranged differently for a specific purpose. You can even save it sorted in a number of ways to different separate files, but keep in mind that when updating time comes, you will have multiple files to update.

Don't forget the columns; they can be sorted as well without jeopardizing the integrity of the information in a record. The fields will just be in a different order in a record.

Since we are going to set up the mailing list in a spreadsheet, we can establish the structure pretty much as we need, to do the job. The specifications include the number of fields to use, what goes into each field and how wide the column for each field should be. Here is a setup that works well for me.

Column	Contents	Width
A	ACCOUNT #	25
B	FIRST NAME	16
C	LAST NAME	16
D	ADDRESS 1	25
E	ADDRESS 2	25
F	CITY, STATE	20
G	ZIP CODE	10
H	TELEPHONE	12

The account number could include a variety of information and it may or may not be printed on the label. A membership or subscription expiration date is a natural use, but you could also code sales activity or other needed data into the field. First and last names are in separate fields to make possible sorting on the last name within the spreadsheet.

Two address lines allow entry of a company name or suite, apartment or box numbers along with street address.

City and state are grouped in one field since mailing sorts are done on ZIP codes rather than city or state. ZIP codes are set up so that state and city automatically group on a ZIP code sort. Note the 10-character ZIP code field that supports the new nine-digit-plus-dash ZIP codes. The 12-character telephone accepts long distance numbers.

The program in Listing 1 prints one-across, one inch high labels on continuous label stock. It will send its output to a printer, or to a disk file for later use, perhaps with a mail-merge program to print personalized form letters.

After clearing 10,000 bytes of string space, the program jumps to Line 100 where field size data is read into the CR(x) array. First the program reads A$ and then CR(x) in a FOR/TO/NEXT loop. The data in A$ is essentially thrown away. We can see why by looking at Line 1000.

```
1000 DATA ACCOUNT#,25,1ST
     NAME,16,LAST NAME,16,
     ADDRESS1,25,ADDRESS2,25,
     CITY/ST,20,ZIP,10,
     TELEPHONE,12
```

Note the data pairs, for example ACCOUNT#,25. This documents a field name and its associated field width. Should you decide that you need 35 characters in the account field, set up your spreadsheet accordingly and then change the 25 to 35 in Line 1000 and the program will properly print your labels.

"City and state are grouped in one field since mailing sorts are done on ZIP codes rather than city or state."

Lines 105 to 140 ask you for information on files to be used and opens these. Where there is a choice like "Printer" or "File," press a P or an F. It does not matter if your CAPS is on or not since the INKEY$ subroutine in Line 1 that gets your response, converts any lowercase characters to uppercase.

Lines 150 and 155 find out if you want the account number or telephone printed. Because only six lines fit on a one-inch label and we should have a blank line between name blocks, you can print either the account or the telephone number, but not both. The account information is printed above the address data while the telephone number prints below.

The real work starts in Line 158 where the total record length (RL) is calculated. In Line 160, an end-of-file test, EOF(1), is made on the input file. In 165, we get a record and test it to see if it contains a full record. If LEN(A$)<RL the record is defective in some way and an error might occur when the program tried to use the record. Unless you made a range mistake in saving your ASCII file, all spreadsheet-produced records should be right. The same cannot be said for files produced in other ways.

From lines 170 through the end of the program take the record apart and print the label. The variable WS$ is used here in conjunction with the subroutine at Line 30. Data is drawn from the record and put into WS$. In Line 30,

it is determined if the printer or file option had been chosen and routes the output accordingly.

Remember that we put first name and last name into different fields? Now we must combine the contents of the two fields. But, first, any spaces that are tacked onto the end of the first name to fill out the field must be stripped off. The subroutine at Line 20 does that. This subroutine keeps looping and dropping one right space from WS$ as long as the right character remains a space. If the field is empty, Line 20 will return with L equal to zero. We will use this later.

Line 175 obtains the first name and calls Line 20 to trim trailing spaces. In Line 180, a separating space and the last name are added to the string. Line 30 is called to output the string.

Lines 185-200 deal with "Address1" and "Address2" fields. Line 20's subroutine determines if a field is empty. If so, it is not output, but the variable EX is incremented and used to print compensating blank lines at the bottom of the label. Line 205 concatonates the city, state field with the ZIP code field, has them output and causes the telephone number to be output if you chose that option.

The program is fairly short. Its logic is possibly more challenging than the BASIC involved for those with some experience in programming.

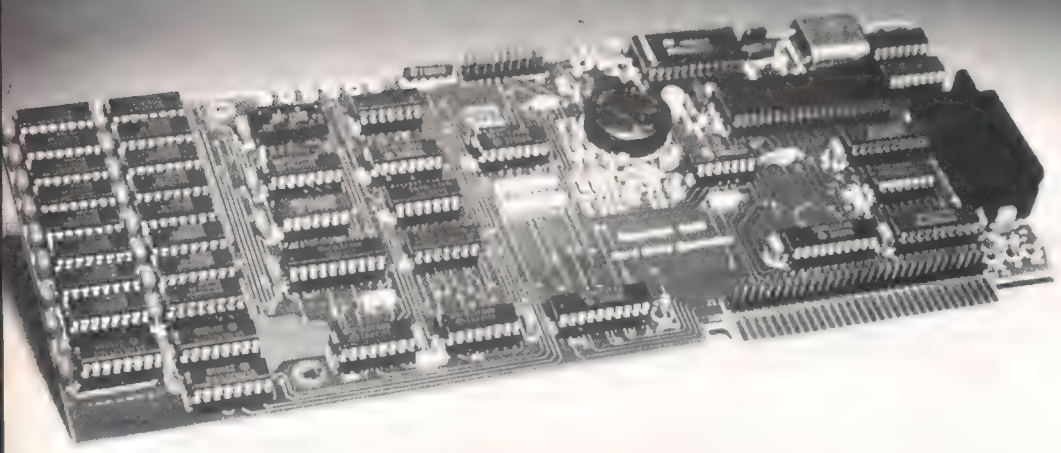
This approach gives you another tool for tailoring spreadsheet printouts. A prime tool has always been to use a word processor or text editor on a spreadsheet ASCII output file. This is particularly good if the output file is close to what you want the final report to be. However, converting a spreadsheet listing to label format in a text editor is too much hard work and just the type work computers are meant to do. □

The listing:

```
Ø CLEAR 1ØØØØ :GOTO 1ØØ
1 I$=INKEY$ :IF I$="" THEN 1 ELSE IF ASC(I$)>96 THEN I$=CHR$(ASC(I$)-32) :RETURN
 ELSE RETURN
2Ø L=LEN(WS$) :IF RIGHT$(WS$,1)=" " THEN IF L-1=Ø THEN L=Ø :RETURN ELSE WS$=LEFT
$(WS$,L-1) :GOTO 2Ø ELSE RETURN
3Ø IF DO=-1 THEN LPRINT WS$ :RETURN ELSE PRINT #2, WS$ :RETURN
1ØØ CLS :FOR X=1 TO 8 :READ A$ :READ CR(X) :NEXT :PRINT TAB(31)"PCM LABEL PRINTE
R" :PRINT TAB(3Ø) "BY RICHARD A. WHITE"
1Ø5 LOCATE 5,3Ø :PRINT "SEND FILE TO:" :LOCATE 7,35 :PRINT "pRINTER" :LOCATE 9,3
5 :PRINT "dISK"
11Ø LOCATE 1Ø,3Ø :GOSUB 1 :IF I$="P" THEN DO=-1 :DO$=" PRINTER " :ELSE IF I$="D"
 THEN DO=1 :DO$=" DISK " ELSE 1ØØ
115 LOCATE 1Ø,3Ø :PRINT "OUTPUT IS TO "DO$
12Ø LOCATE 12,3Ø :PRINT "ENTER INPUT FILENAME" :LOCATE 14,3Ø :LINE INPUT FI$ :IF
 LEN(FI$)>12 THEN LOCATE 14,3Ø :PRINT STRING$ (4Ø,32) :GOTO 12Ø
13Ø LOCATE 16,3Ø :PRINT "OPENING FILE "FI$ :OPEN "I",#1,FI$ :IF DO=-1 THEN 15Ø
135 LOCATE 18,3Ø :PRINT "ENTER OUTPUT FILENAME" :LOCATE 2Ø,3Ø :LINE INPUT FI$ :I
F LEN(FI$)>12 THEN 135
14Ø OPEN "O",#2,FI$
15Ø LOCATE 4,1 :FOR PO =4 TO 2Ø :PRINT STRING$(7Ø,32) :NEXT :LOCATE 4,3Ø :PRINT
"PRINT ACCOUNT FIELD Y/ANY?" :GOSUB 1 :IF I$="Y" THEN AC=1 :LOCATE 6,3Ø :PRINT "
PRINTING ACCOUNT FIELD." ELSE AC=Ø
155 IF AC<1 THEN PRINT :PRINT TAB(3Ø) "PRINT TELEPHONE Y/ANY?" :GOSUB 1 :IF I$="
Y" THEN TL=1 :PRINT :PRINT TAB(3Ø) "PRINTING TELEPHONE" ELSE TL=Ø
158 FOR X=1 TO 8 :RL=RL+CR(X) :NEXT
16Ø IF EOF (1) THEN CLOSE :PRINT :PRINT TAB(3Ø) "COMPLETE" :END
165 LINE INPUT #1,A$ :IF LEN(A$)<RL THEN PRINT "BAD RECORD-- "A$  :GOTO 165
167 IF AC=Ø AND TL=Ø THEN WS$=" " :GOSUB 3Ø
168 WS$=" " :GOSUB 3Ø
17Ø CT=CR(1)+1 :IF AC=1 THEN WS$=LEFT$(A$,CR(1)) :GOSUB 3Ø
175 WS$=MID$(A$,CT,CR(2)) :GOSUB 2Ø :CT=CT+CR(2)
18Ø WS$ =WS$+" "+MID$(A$,CT,CR(3)) :GOSUB 3Ø :CT=CT+CR(3)
185 WS$=MID$(A$,CT,CR(4)) :GOSUB 2Ø :CT=CT+CR(4)
19Ø IF L>Ø THEN GOSUB 3Ø ELSE EX=EX+1
195 WS$=MID$(A$,CT,CR(5)) :GOSUB 2Ø :CT=CT+CR(5)
2ØØ IF L>Ø THEN GOSUB 3Ø ELSE EX=EX+1
2Ø5 WS$=MID$(A$,CT,CR(6)+CR(7)): GOSUB 3Ø :CT=CT +CR(6)+CR(7):IF TL=1THEN WS$=MI
D$(A$,CT,CR(8)) :GOSUB 3Ø
21Ø IF EX>Ø THEN WS$=" " :GOSUB 3Ø :EX=EX-1 :GOTO 21Ø
22Ø GOTO 16Ø
1ØØØ DATA ACCOUNT#,25,1ST NAME,16,LAST NAME,16,ADDRESS1,25,ADDRESS2,25,CITY/ST,2
Ø,ZIP,1Ø,TELEPHONE,12
```

PCM

The Ultimate Expansion for the Model 1000!
TanPak ™ by Hard Drive Specialist



7 Functions Include:
DMA
Serial Port
Memory up to 512K
Clock/Calendar
Printer Spooler
Memory Disk
Expansion Port for Future Options

Features Include:
Gold Edge Cards
Pre-Tested and Burned In
Full Documentation
Supporting Software
One Year Warranty
Expandable to 512K

TanPak ™

The TanPak ™ expansion board has been designed to allow expansion beyond the scope of the standard Model 1000. Seven of the most needed functions/features have been combined into one package using only one expansion slot. Your remaining spaces are left free for future expansion needs.

Seven Function/Features on One Board

512K Memory Expansion

Socketed and expandable to 512K. This is done by two banks of memory using either two 64K increments (128K), or 256K increments (256K or 512K). This allows a total of 640K in the Tandy 1000.

Serial Port

Using the same configuration as the Model 1000 port you are assured of complete compatibility as well as being able to configure it as COM1 or COM2.

Clock-Calendar

Quartz-controlled for a high degree of accuracy, featuring a battery backup.

DMA

The DMA (Direct Memory Access) is used on the Model 1000's first memory card. It increases memory speed and is a must for hard drive operation.

Printer Spooler

Use part of your TanPak ™ memory as a printer buffer. Choose the amount of buffer space you need and stop waiting on your printer.

Memory Disk

Use part of your TanPak ™ memory as a RAM drive. With a solid state drive you can store, retrieve, and sort data quickly and easily.

Expansion System

The TanPak ™ was designed with an expansion port that will allow upgrading with additional features when they become available. Some of the possible features are: a second parallel port, a second serial port, mouse, hard drive port, bubble memory, A/D, and D/A as well as many others.

TanPak ™ 0K . . . . . $349.

Call for pricing on memory modules.

TanPak ™ Secondary

If you already have a Model 1000 memory board and do not wish to replace it, the TanPak ™ Secondary is for you. It features Memory up to 256K, Clock/Calendar, Serial Port, Printer Spooler, and Printer Buffer.

TanPak ™ Secondary 0K . . . . . $249.

TanTel ™

Internal 300/1200 baud Modem for the Tandy 1000. Supported by the DeskMate software package.

TanTel ™ . . . . . $249.

Ordering Information

Use our W.A.T.S. line to place your order via Visa, MasterCard, or Wire Transfer. Or mail your payment directly to us. Any non-certified funds will be held until proper clearance is made. COD orders are accepted as well as purchase orders from government agencies. UPS ground is our standard means of shipping unless otherwise specified. Shipping costs are available upon request. Tandy and Radio Shack are trademarks of the Tandy Corporation. TanPak, and TanTel are trademarks of Hard Drive Specialist, a division of Compukit Corporation.

Hard Drive Specialist
16208 Hickory Knoll
Houston, Texas 77059
1-713-480-6000
Order Line
1-800-231-6671

Megabytes for Mini-Bucks

Tandy 1000 Hard Drives
by Hard Drive Specialist

Shop and compare. Hard Drive Specialist has been building hard drive systems for years and have sold thousands of subsystems to satisfied Radio Shack/Tandy customers. We use the latest state-of-the-art disk drives and controllers. Our drives all use buffered seek logic and plated media to result in almost one-fourth the average access times found on our competitors drives. Plated media results in longer platter life and high resistance to head crashes not found on coated media drives. We utilize high quality construction throughout including heavy duty power supplies and gold edge card connectors. Internal drive systems include an interface card and a half-height herd drive that replaces the top disk drive in both size and power consumption. External units include an interface card, case, power supply and hard drive unit. All units require a memory board with DMA such as the TanPak™ multifunction card, or a Tandy Board, Part #25-1004.

Hard Disk System Features:

- All sizes rated after formatting.
- One secondary drive may be added.
- Error checking and correcting controller.
- Software drivers included.
- Buffered seek drives for improved access time.
- Built in power up diagnostics.
- Plated media for long disk life.
- Gold connectors used throughout.
- 1 year warranty.
- Boots directly from Hard Drive.
- Uses Tandy 1000 MS DOS.

10 Megabyte Internal	**$ 549.**
20 Megabyte Internal	**$ 749.**
30 Megabyte Internal	**$1349.**
10 Megabyte External	**$ 699.**
20 Megabyte External	**$ 899.**
30 Megabyte External	**$1495.**

Ordering Information

Use our WATS line to place your order via Visa, MasterCard, or Wire Transfer. Or, mail your payment directly to us. Any non-certified funds will be held until proper clearance is made. COD orders are accepted as well as purchase orders from government agencies. UPS ground is our standard means of shipping unless otherwise specified. Shipping costs are available upon request. Tandy and Radio Shack are trademarks of the Tandy Corporation. TanPak, is a trademark of Hard Drive Specialist, a division of Compukit Corporation.

Hard Drive Specialist
16208 Hickory Knoll
Houston, Texas 77059
1-713-480-6000

Order Line
1-800-231-6671

We put the finishing touch on our mailing list program this month by bringing everything together in a user-friendly menu

Building a dBASE Menu

By Danny Humphress

For the past two months, this space has been devoted to our first dBASE II programming project — the mailing list program. We have built the three main sections of the system: a program to enter and edit records, one to print a report and one to print mailing labels. Now it's time to bring these three programs together into a "menu-driven" environment.

Menus, as you know, are screen displays of choices for a user to make. In our case, we will want a menu whereby the user can select one of the three programs we have written for the mailing list system. Instead of having to remember the names of the individual files, the user can type something like `DBASE MAIL` and have access to all the related dBASE II programs.

For those of you who may have missed the first two parts of this series, the structure and indexes for the database file appear in Figure 1. Figure 2 is the screen format file and Figures 3, 4 and 5 are the three programs we've written thus far.

Figure 1:

```
STRUCTURE FOR FILE:  C:MAIL    .DBF
NUMBER OF RECORDS:   ØØØØØ
DATE OF LAST UPDATE: Ø8/Ø1/85
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC

ØØ1       NAME         C    Ø3Ø
ØØ2       STREET       C    Ø3Ø
ØØ3       CITY         C    Ø15
ØØ4       STATE        C    ØØ2
ØØ5       ZIP          C    Ø1Ø
ØØ6       PHONE        C    Ø13
** TOTAL **                ØØ1Ø1

. INDEX ON NAME TO MAILNAME
. INDEX ON ZIP TO MAILZIP
```

Building a Menu

As mentioned, we'll want to have selections in our menu for the three programs that we've written. We'll also need a selection to allow the user to exit out of the program and back to MS-DOS. While this seems a simple concept, even experienced programmers, who are used to breaking out of a program in abnormal ways, often forget to add a way out of their program. Imagine going into a grocery store through the "in" door and later, with your arms full of groceries, discovering that the builder had forgotten to put in an "*out*" door!

The mailing list menu program, `MAIL.PRG`, is listed in Figure 6. As we have done in the past, let's go over it line-by-line.

The first four lines, as in the previous four programs, set up the operating conditions for dBASE.

Line six sets up an endless loop between it and Line 12. We need an endless loop here because we want the operations within it to be performed until the user exits from the program.

When the user does exit, line 39's quit command gets them out of the loop, out of the program, and out of dBASE II.

Line eight simply erases the screen to make things nice and tidy for our menu.

Lines nine through 13 display a title and the four menu selections. The screen coordinates were figured so that the menu will be nicely centered on the screen. An easy way of doing this when doing your own programming is to use a word processor that will automatically center text on the screen. With the text centered (make sure your text

> "The conditional on Line 16 is rather lengthy because of the number of valid answers. There is a better way of doing this . . ."

width is set to 80 characters to match the screen size), count the number of spaces from the left edge to the first character or, if the word processor has this feature, look at the column number where each line begins.

Lines 15 through 20 wait for the user to enter a valid response. This type of answer checking loop is a common dBASE programming technique and has been used in the other three programs of the mailing list system. The conditional on Line 16 is rather lengthy because of the number of valid answers. There is a better way of doing this, but this way is easier to understand. Hint: change the line to read do while .not. ANSWER$'1234'.

Lines 22 through 25 check to see if the user has selected menu item Number 1. If they have, the do MAILEDIT command is executed and control is turned over to that program. When that program exits with a return command, control will be returned to this program and execution will resume at the next line. The next line, loop, tells dBASE to go back to the beginning of the do while loop, where the screen will be

Figure 2:

```
@ 1Ø,15 say 'Name:    ' get NAME
@ 12,15 say 'Street:  ' get STREET
@ 14,15 say 'City:    ' get CITY
@ 14,39 say 'State: ' get STATE picture '!!'
@ 14,49 say 'ZIP: ' get ZIP picture '99999 9999'
@ 16,15 say 'Telephone Number: ' get PHONE picture '(999)999-9999'
```

Figure 3:

```
set talk off
set colon off
set confirm on
set bell off

use MAIL index MAILNAME,MAILZIP

erase

do while T
   store ' ' to ANSWER
   erase
   @ 2Ø,25 say 'Select: [A]dd [E]dit e[X]it ';
   get ANSWER picture '!'
   read
   clear gets
   @ 2Ø,Ø

   if ANSWER='A'
      set format to MAILEDIT
      append
      set format to
   endif

   if ANSWER='E'
      store '                    (30 spaces)                    ' to SEARCH
      @ 2Ø,19 say 'Enter name: ' get SEARCH
      read
      clear gets
      store trim(SEARCH) to SEARCH
      find &SEARCH
      if #=Ø
         @ 21,18 say 'Name not found.  Press any key to continue.'
         set console off
         wait
         set console on
         loop
      endif
      set format to MAILEDIT
      edit #
      set format to
   endif

   if ANSWER='X'
      use
      return
   endif

enddo
```

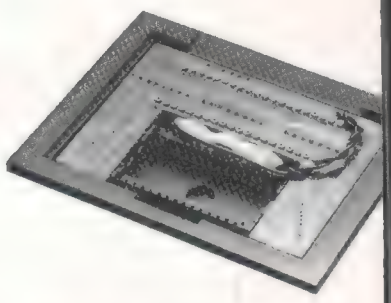
Figure 4:

```
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH M=10,W=132
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: Mailing List Report
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) N
COL       WIDTH,CONTENTS
001       30,NAME
ENTER HEADING: <Name
002       30,STREET
ENTER HEADING <Street
003       15,CITY
ENTER HEADING: <City
004       2,STATE
ENTER HEADING: <St
005       10,ZIP
ENTER HEADING: <ZIP Code
006       13,PHONE
ENTER HEADING: <Telephone
```

Give your Model 100

128K RAM

Installs as easily as plugging in a socket



Software included, transfers from bank to bank. Works like main menu! Includes powerful RAM Basic that lets programs store and access data from any other bank.

PCSG says: Satisfaction guaranteed or your money back within 30 days

As amazing as it seems you can upgrade your Model 100 to 128K of RAM in just 60 seconds.

It comes to you right out of the box looking just like the picture. You just open the little compartment on the back of your Model 100 with a quarter and it just pushes right into place. You can then put the cover back in its place.

You then have 4 banks of RAM of 32K each. The additional three banks also work just like your Main Menu.

You push a function key and you are in the second bank. Push again and you are in third, again, then fourth. Press it once again for your original bank.

It has its own built-in NiCad battery that recharges right from the Model 100 and its guaranteed for a full year.

What is really great is that you can copy a file from one bank to another with just a function key.

Each bank is like having another Model 100, and all the built-in programs as well as any snap-in ROM programs appear in all four banks and work the same way. Your widebar cursor moves from file to file and you access any file or run any program just by pressing ENTER.

What lets you copy any file from one bank to another is a snap-in ROM from PCSG called RAM+, that comes at no extra charge. It just pushes right into the little socket in that same compartment with the 96K expansion unit.

Not only does this firmware let you copy a file from bank to bank, but you can make a copy of any file within the same bank instantly with a function key. Great for Lucid spreadsheets!

Copy a file from bank to bank with a function key

You can also rename a file, or kill any file with just a function key. Plus you can do a whole lot of other useful things like setting the date, day and time with function key ease. You even have a function key that lets you use non-Radio Shack printers without having to make those tricky dipswitch settings.

RAM+ lets you cold start any one of your banks without affecting the other three. That means that anytime you want you can clean out a bank's entire memory, but leave intact all the files in the other banks.

What is also fantastic is that you don't have to have the ROM in place to use the additional RAM. Whenever you take out the snap-in ROM it leaves behind a tiny machine code program that lets you switch from bank to bank just by pressing ENTER.

This lets you use your ROM socket to snap-in other ROMS like LUCID spreadsheet, WRITE ROM text processor, or DISK+ ROM file transfer program, and use them in any or all four banks. All of these, by the way, are available from PCSG.

When you are ready to copy a file from one bank to another or use any of the other fantastic functions we talked about you can just snap the RAM+ ROM back into place.

Everybody that has this 128K system in their Model 100 is so excited, because it gives them four times the capacity and all banks work just like the Main Menu.

And what has made a lot of people happy is that the system bus, located in the same compartment, is left free for you to plug in a DVI or the Holmes Engineering/PCSG portable disk drive.

The ability to copy a file from bank to bank instantly with a function key, plus all of the other features make this RAM extension truly an engineering masterpiece.

Some people hesitate when they think of installing something, and then others are skeptical that any additional hardware could be as good as the Model 100 itself. That's why we sell these 96K expansions on a 30 day trial. Simply return it within 30 days for a full refund if you are not satisfied. Priced at $425. MC VISA COD.

1-214-351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd. No. 207, Dallas, Tx. 75229

VISA / MASTER CARD / AM. EXP. / C.O.D.

cleared and the menu will be displayed once again.

The commands in lines 27 through 30 and 32 through 35 perform the same functions as the previous `if/endif` condition. The same functions, that is, except that they call the other two programs.

If the person selects Number 4, it will be detected in Line 37, the screen will be erased, and dBASE will quit.

Question: When will the flow of this program ever reach the `enddo` in Line 42? The answer is "*never.*" The loop in lines 16 through 20 will not allow the user to enter anything other than '1', '2', '3' or '4', thus, the answer will be trapped by one of the `if/endif` conditionals. Each `if/endif` has either a `loop` command, which returns program flow to the top of the loop, or a `quit` command which causes dBASE to exit back to MS-DOS.

Congratulations! You have written your very first dBASE II program . . . well, with just a little help from this end. With this mailing list program as a foundation, you should be able to build something you can call your own. □

Figure 5:

```
1  set talk off
2  set colon off
3  set confirm on
4  set bell off
5
6  erase
7
8  @ 10,20 say 'Preparing to print mailing list report.'
9  store ' ' to ANSWER
10 do while (ANSWER<>'Y' .and. ANSWER<>'N')
11    store 'N' to ANSWER
12    @ 12,27 say 'Do you want to continue? ' get ANSWER picture '!'
13    read
14    clear gets
15 enddo
16 @ 10,0
17 @ 12,0
18
19 if ANSWER='N'
20    return
21 endif
22
23 store ' ' to ORDER
24 do while ORDER<>'N' .and. ORDER<>'Z'
25    store 'N' to ORDER
26    @ 12,22 say 'Sort the list by [N]ame or [Z]ip? ' get ORDER picture '!'
27    read
28    clear gets
29 enddo
30 @ 12,0
31
32 if ORDER='N'
33    use MAIL index MAILNAME
34 else
35    use MAIL index MAILZIP
36 endif
37
38 @ 12,34 say 'Printing...'
39 set console off
40 report form MAILLIST to print
41 set console on
42
43 use
44 return
```

Figure 6:

```
1  set talk off
2  set colon off
3  set confirm on
4  set bell off
5
6  do while T
7
8     erase
9     @ 9,30 say 'Mailing List Program'
10    @ 11,25 say '1.  Enter or Edit Records'
11    @ 12,25 say '2.  Print Mailing List Report'
12    @ 13,25 say '3.  Print Mailing Labels'
13    @ 14,25 say '4.  Exit'
14
15    store ' ' to ANSWER
16    do while ANSWER<>'1' .and. ANSWER<>'2' .and. ANSWER<>'3' .and. ANSWER<>'4'
17       @ 16,28 say 'Your selection please: ' get ANSWER picture '9'
18       read
19       clear gets
20    enddo
21
22    if ANSWER='1'
23       do MAILEDIT
24       loop
25    endif
26
27    if ANSWER='2'
28       do MAILLIST
29       loop
30    endif
31
32    if ANSWER='3'
33       do MAILLABL
34       loop
35    endif
36
37    if ANSWER='4'
38       erase
39       quit
40    endif
41
42 enddo
```

PCM

Changes your Model 100 into a totally different computer with capability you never thought possible.

Lucid™

on Snap-In™ ROM

Infoworld rated Lucid's performance "excellent"

PCSG says "Satisfaction Guaranteed or your money back within 30 days!"

LUCID© is here now. It is on a ROM cartridge that snaps into the compartment on the back of your Model 100. It takes no memory to load and no memory for operating overhead. That means you have the full 29.6k bytes free to store your data.

First, LUCID© is memory conserving. It will let you build a large spreadsheet – 255 row by 126 column capacity. You build huge spreadsheets in your Model 100's RAM that could consume 80 to 100K on a desktop computer.

Secondly, LUCID© is fast. LUCID© is so rapid, a 36 column corporate financial statement took less than 4 seconds to calculate.

Thirdly, LUCID© has features you won't find in most other spreadsheets. For example, when you type a label (text) it will cross column boundaries; in other words when you type a label or title it will appear as you type it irrespective of column or width. LUCID© also allows you to set column widths individually, and of course LUCID© has insert row and insert columns, as well as other standard features. LUCID© even lets your formulas refer to cells in other spreadsheet files.

Further, LUCID© has what no other spreadsheet has: Cut, Copy, and Paste. It uses the same keys as Cut and Paste in TEXT, but here's the difference: it takes all the formulas with it when you paste and they all automatically recalculate with the entire sheet.

And here is what is really amazing. You can copy or cut from one spreadsheet and paste into another spreadsheet or even a TEXT file.

LUCID©supports all BASIC math functions as well as Log, sine, cosine, tangent, exponentiation and other sophisticated math functions.

LUCID© has so many features that you will say "this is what I need in a spreadsheet", such as automatic prompting of an incorrectly typed-in formula showing just where the mistake was made.

LUCID© has expanded "go to" functions that remember and produce a windowing capability.

But perhaps most remarkable is that LUCID© is not only a spreadsheet but a program generator as well. First, LUCID© lets you protect all cells against entry or change, and then unprotect just the cells you want for someone else to use as input fields.

LUCID© will not only process values, but text input as well so that the facts other than numbers can be responded to. LUCID© has the ability for you to refer in a formula to cells containing words. This feature combines with the capacity of doing "if then" statements that work by doing table lookups against even massive X/Y charts of text or numerical information. You can produce a program that responds to inputs with no programming knowledge whatsoever.

You can prepare a report section in your spreadsheet with instructions to your user for printout, and they can produce a personalized printout that responds to their input. All your formulas and tables that did the calculations and provided the facts are invisible to that user. LUCID© is useful for doctors for patient questionnaires, troubleshooting technicians, purchase clerks, people doing job quotes, stores for customer workups, insurance agents and anybody who needs to process specific facts and numbers to produce a report based on those responses.

LUCID© comes with a manual that explains not only the characteristics of LUCID©, but will train you how to use a spreadsheet even if you have never seen one before. You are shown how to do budgets, forecasts, breakeven analysis amortizations and many other types of personal and business reports and calculations.

User friendly is such an over-used term in this industry, but a typical comment has been "I have never seen a spreadsheet that does so much, and yet LUCID© is so much easier and faster to use."

LUCID© is a result of a most exhaustive developmental effort in which PCSG's objective was to develop a spreadsheet that was better than the state-of-the-art. We are so pleased because LUCID© provides for the Model 100 spreadsheet capability you cannot equal on a desktop computer.

LUCID© is, in our opinion and that of those who have examined it, a breakthrough. We sell it on a 30 day trial. If you are not completely satisfied, return it within 30 days for a full refund. Priced at $149.95, on snap-in ROM. MasterCard, Visa or COD.

First with software for the Model 100

PORTABLE COMPUTER SUPPORT GROUP

1-214-351-0564

11035 Harry Hines Blvd., #207
Dallas, Texas 75229

Framework: An Integrated Power House

Software 1000/1200/2000

Lotus 1-2-3 was the first integrated spreadsheet, database and graphics software for IBM PC compatibles. I found each of those three features of *Lotus* to be rather weak. The spreadsheet wasn't much better than *VisiCalc* at a time when hard-nosed computer people (such as myself) were starting to be spoiled by "second generation" spreadsheets such as *Multiplan*. The database had all the muscle of a sleeping baby. At least it could be said that the graphics features were adequate.

Of course, the main advantage of this type of software is the fact that you need only one program rather than several; however, none of these integrated products will handle huge amounts of data. Besides, most people *I know* who bought *Lotus* got it for only one of its features. An employee in a key position of one of the largest corporations in the country told me he chose *Lotus* to be used merely as a spreadsheet company-wide! He could have spent a lot less money and gotten a much more powerful (as well as more user-friendly) program for the one application he needed.

Now there is an integrated software package that addresses some of the shortcomings of earlier integrated programs: *Framework* from Ashton-Tate, the same people who gave the world *dBASE II*, I must admit that since my opinion of *dBASE* is none too high, I wasn't expecting to like *Framework* very well. However, after I worked with the package, its advantages over other products of its kind were readily apparent. There are many features in *Framework* that *Lotus 1-2-3* doesn't possess unless one goes to the extra expense of buying *Symphony* (not the least of which is word processing). There are also some unique features in *Framework*'s little bag of tricks.

That doesn't mean there aren't some negative points to *Framework*, but I'll talk about those later.

The best news first. Unlike *Lotus 1-2-3, Framework* is worth buying just for its spreadsheet capabilities. How many spreadsheets do you know of that have built-in functions to figure loan amortization payments, permutations, currency conversions, modified internal rate of return, *all* major inverse trigonometric functions, triangle side solving and variance? It's heaven for any number crunching business or scientific user! And I though *Multiplan* was awesome.

What gives the spreadsheet its clout is its ability to borrow functions from *Framework*'s intrinsic language called Fred. Yeah, I laughed too when I first heard the name; it's an acronym for FRames EDitor. What is really interesting is that you can use Fred without the spreadsheet or any other application in *Framework*. Fred can be a standalone language for program development and as far as function repertoire is concerned, probably outranks both BASIC and PASCAL.

As with any spreadsheet, the more memory your PC has, the more cells you can fill. On a 256K machine you can fill 522 cells; at 384K you'll have 1,860; with 512K you'll get a maximum of 3,197 and 640K will permit up to 4,534.

Just as in *Multiplan*, you don't have to enter formulae in terms of cell coordinates; rather, you can have English expressions. Instead of saying "profit is B2-C2," you can tell the spreadsheet that it equals "sales — cost." From the delving I have done so far, I can't think of any feature of *Multiplan* which can not be implemented on *Framework*.

The word processing feature is fairly impressive. Unlike many word processors, when you want boldface text it not only appears as boldface on paper, but also on your monitor's screen. The same applies to underlining and italics. Also, since you can incorporate a spreadsheet (as well as Fred functions) into your documents, you essentially have a word processor with mathematical capabilities.

One major drawback of the word processor is that there are no provisions for superscripting, subscripting or spelling checking.

One expects a spreadsheet to be limited by memory; not so a database. With a true database the limitation to the size of a file is the amount of disk space available on the computer, not the computer's RAM capacity. Unfortunately, *Framework*'s filing program can produce files no larger than the

amount of free RAM memory there is in your system. For instance, you can only store 76 records of 10 fields with 10 characters per field on a 256K computer. For this reason I refer to this part of the package as a filer rather than a database. Unless your record storage requirements are very light, you probably won't find *Framework*'s filer function very useful.

Framework's graphics resolution is rather crude compared to *Lotus 1-2-3*. Both packages produce plots of the bar, linear and pie varieties.

I've never heard of an outliner before. To be honest, I was initially a bit leary as to the practicality of this *Framework* component. Who needs a separate utility to write outlines for written works? Why not use the word processor? But the outliner goes a step further than that. For each topic you put into your outline it sets aside a block of memory (a frame) into which you can later type specific text concerning the topic. Thus, your outline can actually be gradually turned into the finished work.

Mite, the communications part of *Framework*, is as good as any standalone smart telecommunications program I've seen. Uploading, downloading, auto-answer, auto-dial, and XMODEM protocol are among the many functions which can be invoked. However, I did not notice any radically new or unheard of features. *Mite* requires a computer with at least 384K.

Documentation is extensive and consists of a tutorial manual and a reference manual. These two books together comprise more than 1,000 pages that are, on the whole, fairly readable.

In closing, I think it's important for me to say, that there are too many features in *Framework* for any review to fully cover all of them. Whether *Framework* is suitable for you depends upon what you need done. However, I can categorically state that *Framework* is one of the more superior works of its genre.

(Ashton-Tate, distributed by Tandy Corp., One Tandy Center, Fort Worth, TX 76102, $695)

— Rick Boozer

Software **1000/1200/2000**

Mlink Found to be a Flexible Communications Package

Corporate Microsystems has developed a very reputable package that will work on all three of Tandy's MS-DOS computers: the Tandy 1000, 1200 and 2000. At first, I doubted that it could live up to such high standards, but I have been pleasantly surprised. *Mlink* has taken the needs of most microcomputer installations at heart, and has created a package that at a glance can satisfy almost all typical communication requirements.

Mlink does transcend well beyond the typical communication packages be-

cause of its flexibility and expandability. You, fortunately, aren't limited to the access of bulletin boards, nor is the system so confusing that accessing boards would be the only thing you would use it for.

Mlink is very easy to install. The only obstacle that I encountered was when I tried to use the package with Tandy's DC-2212 modem. Hopefully, for truly faithful Tandy users, Corporate Microsystems will come out with some documentation on how to use this particular modem. When I finally located a Hayes 1200-Baud Smartmodem, I was able to fire it up without a hitch.

There were no complicated installation steps to follow. I simply copied the programs to the hard disk, typed MLINK and was on my way.

Mlink's manual was very concise in its explanations on how to use the software. I followed the manual step by step using *Mlink's* online mode and discovered that it was a snap to move from one option to the next. If and when I ran into any problems, I had ready access to a "help" screen via the on-screen menus.

Once I was sure the system was configured properly (Baud rate, parity, etc.), I was ready to dial my first number. Immediately I achieved success. This is usually enough incentive to peak my curiosity, so I investigated further. I was able to set the auto-dial function so it would retry the number up to eight times before hanging up.

For computer power-users *Mlink* becomes very tempting because of its transfer file option. I experimented with this option by hooking up two computers in the office to see if they would cooperate and successfully send information to one another. This is often at best a very difficult task necessitating a new growth of hair. This option worked smoothly and I was able to send files from one computer to the other. During the transfer process, *Mlink* traced the transfer process by displaying the action taking place.

I could also opt to send a list of specified files from a transfer list instead of one at a time by using a wild cards option.

Files can also be transferred in the verbatim mode, by-passing all error checking and transferring the data as is. Unless you set the toggle off, *Mlink* will display the contents of the file on the screen as it is being transferred.

As I played with *Mlink* I began to become more confident. I didn't want to thumb through the menus. *Mlink* allowed me to do this with its "brief mode." I could go directly to the task that I wanted without fishing for the correct menu.

Mlink supports another option that makes it extremely powerful, "scripts." These can be used for simple tasks such as logging into another system or for tasks as complex as having a computer start itself at a certain time of the day, copy a series of files, delete a file or two, leave a message and log off without intervention.

The script language provided makes the possibilities for communication very exciting. The script language allows someone to get into *Mlink* and change virtually all of the system's defaults. File I/O, keyboard and screen I/O can be controlled with the script language.

The remote script allows remote access to your computer. By doing this you can have remote access to do file

PERRY COMPUTERS

CALL
1-800-248-3823

COLOR COMPUTERS		LIST PRICE	OUR PRICE
26-3127	64K Extended Color Computer 2	$219.95	$175.00
26-3134	16K Standard Color Computer 2	119.95	102.00
26-3129	Thinline Disk Drive O for Color Comp	349.95	290.00
26-3018	Extended Basic Kit	39.95	36.00
26-3030	OS-9 with Editor Assembler	69.95	59.50
26-3012	Deluxe Joystick	29.95	25.00
26-1208	CCR-81 Recorder	59.95	50.00
VIP	Integrated Library	149.95	139.00
VIP	Writer	69.95	59.00
VIP	Calc	69.95	59.00
Telewriter	Disk		59.00
Botek	Interface		57.00
TANDY 2000			
26-5103	Tandy 2000 Two Disk	$1599.00	$1225.00
26-5104	Tandy 2000 One Disk & 10 Meg HD	2499.00	1885.00
26-5111	Monochrome Monitor	199.95	165.00
26-5112	Color Monitor	599.95	510.00
26-5140	Hi Res Graphic Monochrome Board	299.95	255.00
26-5141	Hi Rec Color Graphics Chips	124.95	106.00
26-5160	Internal 128K Memory Board	179.00	153.00
26-5300	Lotus 123	495.00	420.00
26-5311	Microsoft Multiplan	195.00	166.00
26-5320	Framework	695.00	590.00
26-5352	dBase II (Data Base)	595.00	420.00
26-5330	Multmate	249.00	205.00
MODEL 200 COMPUTERS			
26-3860	24K Model 200 Portable Comp.	$999.00	$725.00
26-3866	24K RAM Mem. Exp. Chip for 200	249.95	210.00
26-3804	AC Adaptor	5.95	5.00
26-3805	Accoustic Coupler	39.95	34.00
26-3816	8K RAM Memory Expansion Chip	119.95	95.00
26-1409	Printer Cable	14.95	12.70
26-1410	Modem Cable	19.95	17.00
26-3809	Briefcase	49.95	42.50
26-3811	Soft Carrying Case for 100/200	39.95	34.00
26-1183	Bar Code Reader	99.95	85.00
26-3806	Disk Video Interface	499.00	415.00
26-3829	Multiplan ROM for 100	149.95	127.50
MODEL 4 COMPUTERS			
26-1070	Model 4D 64K 2 Drive Computer	$1299.00	$ 995.00
26-1122	64K Memory Expansion	69.95	65.00
26-1127	Model 4 Internal Drive 1	399.95	385.00
26-1123	Model 3 to Model 4 Upgrade Kit	399.00	385.00
26-1084	Model 4P Modem Board	149.00	95.00
26-1085	Model 4P Travel Case	24.95	21.25
26-1134	Model 4 Hard Disk Kit for 15meg	79.95	67.95
26-1530	Multiplan Model 4	199.00	169.00
26-1595	Super Scripsit Model 4	199.95	169.00
26-1635	Profile 4 Model 4	249.95	200.00
26-2216	CP/M Plus Model 4	149.00	127.00
26-2231	Double Duty Utility	69.95	58.00
MODEL 12 & 6000 COMPUTERS			
26-4005	Model 12 80K 2 Drive Computer	$3999.00	$1350.00
26-6021	Model 6000 512K 2 Drive Comp.	4499.00	3125.00
26-6022	Model 6000 512K 15Meg HD	5499.00	3795.00
26-6014	Model 12 to 6000 Upgrade	1595.00	1325.00
26-6015	Model 6000 512K Memory Board	1095.00	905.00
26-6019	Model 6000 256K Memory Kit	199.95	185.00
26-6017	Card Cage for Model 12	199.00	169.00
26-6052	DT-100 Data Terminal	795.00	600.00
26-4155	15 Meg Hard Drive Primary	1995.00	1355.00
26-4171	35 Meg Hard Drive Primary	2995.00	2545.00
26-4157	Installation Kit for Primary HD	349.00	295.00

All prices and offers may be changed or withdrawn without notice. Advertised prices are cash prices. C.O.D. accepted ($10.00 charge per carton on C.O.D. Call for further C.O.D. information.) M.C., Visa, AX, add 2%. All non-defective items returned will be subject to 10% restocking fee. Defective items require return merchandise authorization. Call for R.M.A. Number before returning. Delivery is subject to product availability.

RADIO SHACK PRINTERS		LIST PRICE	OUR PRICE
26-1276	DMP-105 Dot Matrix Printer	$ 199.00	$ 169.00
26-1275	TRP-100 Portable Thermal Printer	299.00	230.00
26-1280	DMP-130 100 cps Tri. Mode Printer	349.00	285.00
26-1268	CGP-220 Color Ink-Jet Printer	699.00	595.00
25-1257	DWP-220 Daisy Wheel	599.00	460.00
26-1277	DMP-430 24 Wire Matrix Printer	899.00	745.00
26-1270	DWP-510 43 cps Daisy Wheel	1495.00	1225.00
26-1274	DMP-2100P 24 Dot Wire Matrix	1995.00	1650.00
26-1279	DMP-2200 HiSpeed Matrix Printer	1695.00	1440.00
26-1269	PT-64 Printer Controller	249.95	210.00
26-1498	SW-302 Printer Switch	119.95	100.00
26-1477	Auto Sheet Feeder for DWP 510	499.95	420.00
TANDY 1200			
25-3000	Tandy 1200 One Disk & 10 Meg HD	$1999.95	$1525.00
25-3010	VM-3 Green Monitor	219.95	185.00
26-3212	CM-2 Color Monitor	459.95	390.00
25-3040	Monochrome Display Adapter	219.95	185.00
25-3043	Graphics Display Adapter	299.95	255.00
25-3044	Graphics Master	695.95	540.00
25-3061	Captain Multifunction Board	599.95	475.00
25-3130	MSDOS/BASIC	89.95	76.50
25-3170	Wordstar Professional	395.95	335.00
25-3160	Framework	695.95	590.00
25-3161	PFS File	140.95	119.00
25-3190	dBase III	695.95	590.00
TANDY 1000			
25-1000	Model 1000 128K 1 Disk Drive	$ 999.95	$ 775.00
25-1001	Model 1000 384K 10Meg HD	1999.00	1525.00
25-1004	Memory Exp 128K 1000/1200	239.95	205.00
25-1005	Disk Drive Expansion 1000	199.95	170.00
25-1006	RS-232C Interface 1000/1200	99.95	85.00
25-1007	Hard Disk Control Board	299.95	255.00
25-1013	Modem 1200 Option 1000/1200	299.95	230.00
25-1025	10 Meg Hard Disk	699.95	559.00
25-1501	MS-DOS Reference T-1000	34.95	29.00
25-1502	BASIC Reference T-1000	34.95	29.00
26-3211	VM2 Green Monitor	149.95	125.00
26-3212	CM2 Color Monitor	459.95	390.00
EPSON/COMREX PRINTERS			
20001015	Homewriter 10	$ 288.00	$ 230.00
20001025	LX-80 Dot Matrix Printer	299.00	250.00
20001035	FX-85 Dot Matrix Printer	499.00	385.00
20001040	JX-80 Color Dot Matrix Printer	699.00	485.00
20001050	HI-80 4 Pin Plotter	599.00	390.00
20002010	RX-100 Dot Matrix Printer	499.00	400.00
20002030	FX-185 Dot Matrix Printer	699.00	525.00
20003010	LQ-1500 18 Pin Head with Intf	1495.00	1095.00
20003510	CR-IIE Comrex Com Riter	499.00	370.00
20001510	FX-80 Tractor Feed	39.95	34.00
20004080	NLQ Parallel Intr	209.95	175.00
20001515	LX-80 Tractor Feed	49.00	40.00

For Technical Questions and Information on our complete line of computer accessories and current prices.

CALL 1-517-625-4161
FOR ORDERS ONLY CALL 1-800-248-3823

124 S. MAIN ST, PERRY, MICH. 48872

operations and to transfer files. With remote Script you have the option to control logon, read files and directories, create, erase and rename files.

Other scripts that have been supplied by *Mlink* include: "EMS" — a simple electronic mail system that allows you to create and edit mail messages with other computers running EMS. "RUN" gives the user the ability to run programs while still on line. "SCHED" will perform a script at a scheduled time of the day in order to take advantage of less expensive telephone time.

With my superficial examination, I was very impressed with the ease of use, the well written documentation and the flexibility of *Mlink*. This could be the only communications package that a user would need. If you are either looking for a package for simple communication tasks, or a sophisticated package to handle the needs of a client, *Mlink* will probably be worth your future attention.

(Corporate Microsystems, Inc., P.O. Box 277, Etna, NH 03750, 603-448-5193, Development System $250, Run-time $195)

— Lester Blank

Software 1000/1200/2000

Etch-a-Mouse Found to be a "Fun Toy"

Every now and then you look at a package that you can't help thinking has a lot of potential if only the authors had done . . . This is exactly the case with *Etch-a-Mouse*. The package is a good start of a program but is just that, a *start*.

Etch-a-Mouse is a drawing program that utilizes the Digi-Mouse and the outstanding color of the Tandy 2000.

The documentation consists of thirteen pages stapled together. There are several minor errors and ommissions in the documentation, but anyone with a little experience with MS-DOS should be capable of figuring it out. The documentation for the most part is fairly clear and accurate.

You initiate the package by executing a batch file. The file copies the program files onto your disk and also sets up a AUTOEXEC.BAT file for future use. The batch file also establishes a new prompt character. After the initial start up procedure you are shown a menu consisting of two options, setup colors and run program. If you choose the setup option, eight open blocks and eight color blocks are displayed. The idea is to use the mouse to choose eight colors out of the sixteen that are available for the Tandy 2000. You accomplish this by moving the cursor to the color you want in the bottom row and clicking the right button on the mouse to pick up your chosen color. You then deposit it in one of the open blocks in the top row; the left button on the mouse switches in the alternate shade, thereby giving you the choice of the sixteen different colors. This procedure can be avoided by using the default colors which are also the same colors for Tandy's CGP-220 Ink-Jet Printer. You have to be careful which colors you pick during the setup process because once you're in the drawing mode there isn't any way of changing them. After you have chosen your colors you are shown a title screen that is drawn painstakingly slow.

You then enter into the drawing mode. There will be two blocks in the lower left-hand corner of the screen; one block tells you which color you have selected by clicking the left mouse button. The other block tells you if the color has been turned on by clicking the right mouse button. The mouse ratio of screen to table-top movement has been preset to an area of approximately a sheet of paper and is non-adjustable. There is an on-line help facility, although it did not work. A call to Soft Horizons indicated that the bug had been corrected. I also asked them if they are planning to revise the program to allow you to be able to insert text or if they were planning to provide any provisions to allow you to load a drawing into a basic program. They said that, "there were not any revisions planned at this time."

The program has several commands available for drawing, there is a line command, box command and circle command. There is also a fill command and paint command. There is a command called "fat mode" that allows you to determine your line thickness. This is accomplished by drawing a series of circles one after another; the problem with this is that if you move the mouse too fast the line is not solid. Another problem with fat mode is that you can not use the line command to draw a line from one point to another. There are also commands to flip the cursor up or down and to save or recall your work from disk, provided you have 96K available for each drawing. The drawing commands work by the user's marking a starting location for a line and by moving the mouse to the end location of the line. The line is then drawn between these two locations.

The first location is shown as coordinates at the bottom of the screen, however the end location is not shown. You therefore have to guess to be able to draw a straight line.

The circle command that is mentioned in the instructions, although not described, apparently works by your marking the center and outside radius of the circle.

You have the ability to save and recall your work, but if you try to recall a drawing and forget the filename, you get stuck in the recall command with no way out but to re-boot your system.

The program for the most part uses lowercase letters except for fat mode which uses uppercase. This may not seem like a problem until you switch to fat mode and forget to hit the shift key and enter a lowercase 'f'. This will access the fill command, fill your drawing with some color and render it useless. This can be very frustrating if you have been working on a drawing for several hours.

The disk includes two demonstration drawings that are quite amusing.

The Bottom Line

Etch-a-Mouse is easy to use once you get through the initial copying and color setup routine. My 6-year-old son had no problem using the program, and thought it was a fun toy. However, for any type of serious business use I can not recommend it. There is no provision for inserting text into a drawing nor for adding any drawing you have made into an existing program. *Etch-a-Mouse* has a lot of potential but at this point, it is not much more than a fun toy.

(Soft Horizons, Rt. 1 Box 432, State Hwy. 83, Cape May Court House, NJ 08210, $39.95)

— Tim Frost

Project Scheduler Helps You Follow the Critical Path

For those of you involved in project development, the song "The Long and Winding Road" has a special meaning. Often it seems that in the development of a complex project you can't get from here to there.

My job involves the development and introduction of new products. Since they are to be sold as consumer electronic products, the list of jobs that have to be completed prior to introduction is extensive. Without careful planning, the product ends up ready to ship without packaging or an instruction manual.

To assist Model 100 owners in project development, Tandy has released a program titled *Project Scheduler*. Designed to help you maintain control of even the most complex project, you divide a large project into smaller individual tasks with specific time tables and interdependencies. These jobs define all the activities necessary to complete a project.

You can choose days or weeks as the time unit measurement for each project. Each project can have a maximum of 49 jobs, and every job can have up to 99 time units in length. Every job can be defined with as many as four prerequisite jobs.

Using all this information, *Project Scheduler* calculates the final completion data and highlights "critical path" jobs. The entire project schedule can be printed in either graph or table form for easy distribution and record keeping.

After the project has been entered, changes, additions, deletions and insertions can be made. The program automatically recalculates the schedule, revising the critical path and completion date accordingly.

To use this program, a Model 100 with a minimum of 24K of memory is necessary. Tandy states that the program can be saved in memory if you have 32K; however, unless you are willing to dedicate your Model 100 to this project, it is more sensible to load the program and the file or files pertaining to a particular project, update or print your current status, save the data to tape and kill the program and files, thus freeing up the memory of you computer for other uses.

Project Scheduler has two sets of functions. These are listed at the bottom of the project schedule graph screen. You have the option of selecting the various functions either by pressing the function keys or the first alpha letter of the desired function.

The functions available are:

Add: Add a job to the project.
Insert: Insert a job in the project.
Project: Display the project status.
Update: Update the job details.
Move: Move the position of a job in the schedule graph display.
Delete: Delete a job from the project.
Save: Save the file to a cassette tape.

To activate the second set of function keys, the TAB key is pressed. These commands are:

Complete: Mark completion (full or partial) of a job.
Graph: Print the schedule graph.
Table: Print the schedule table.
Prereq: Print a prerequisite report.

On both function key sets, F8 returns the user to the menu.

Because of the size limitations of the display screen, the graph for the project will overflow the display screen. What you see is a section of the graph (a window) much like one would see when using a spreadsheet. You use the arrow keys to move the window around the graph, or you can print the entire graph to analyze the various jobs, when they start and how they interrelate with each other.

Once *Project Scheduler* is loaded, separate files are created for each

project. The 30-page users manual walks the new user through the operations necessary to create a schedule. Although this program is useful in graphically illustrating the time lines in a project, it does not know the jobs which have to be involved in your project. To get maximum use from the program, some homework is necessary prior to using the program.

The accuracy of the resultant graph is only as accurate as the data fed into it. If packaging is going to take six weeks, the Model 100 is not smart enough to correct you if you optimistically enter four weeks. Nor will it catch you if you try to complete the packaging before a prototype is completed. It does not test your logic, only calculates the combined time line and shows the results.

I was pleasantly surprised to discover that this program was available for the Model 100. This type of program has been available for the desk-top business computers, but the advantage of having it available for the 100 is the simplicity of operation and the advantage of portability. There are many times when I am away from my desk and would like to review a project's status, or determine the need of overtime work premiums to advance the schedule of a project to meet a deadline.

In project management one quickly discovers that the addage, "time is money" is accurate. If you want to cut time, add money. If the magic silicon genie were to grant me one wish for this program, it would be the ability to add the cost of each job of the project with the totals displayed. As the job is completed it would indicate if the project is coming in over or under budget, and let me analyze the effects of rush charges vs. the time line.

For my purposes, I have found *Project Scheduler* to be a very usable program. Both the graphed and the printed reports are very readable even by those not familiar with the program. There is something so official about seeing a report printed on a dot matrix printer. The impact of the graph showing completion beyond the due date is great!

If you are involved with projects requiring multiple tasks which are interdependent on each other for completion, *Project Scheduler* is worth a trip to your local Radio Shack Computer Center for a look. It may save you some unpleasant surprises in the future.

(Tandy, available in Radio Shack Computer Centers nationwide. $39.95 requires 24K)

— Bruce Rothermel

Software **1000/1200/2000**

Payroll Made Simple With *Payday*

As any small business owner readily knows, maintaining a payroll consumes precious bookkeeping time. Less time spent on the payroll and more spent on production or service means more profit. To help streamline the whole payroll process, Heath Research of Virginia Beach, Va., has introduced *Payday*.

Payday is an automated payroll system specifically designed for small businesses. It will run on any MS-DOS or PC-DOS computer with a minimum of 256K RAM and two disk drives or a hard disk. The program is completely written in BASIC.

Because the program is designed for the small business, there are four qualifications, as stated in the manual, that must be met in order for *Payday* to be useful to you:

1) "Your firm writes paychecks on a weekly, bi-weekly, semi-monthly or monthly basis."

2) "Your company does not pay any employee $100,000 or more in any individual pay-period."

3) "Your company spends less than one million dollars on total gross wages for any individual pay-period."

4) "Your firm's total yearly payroll is less than ten million dollars."

The above stipulations should satisfy the requirements of many small businesses — a definite plus for the program's applicability.

Payday is a menu-driven package. When the program is first booted, the main menu, called "Central Control," is displayed with five options:

```
(CP) COMPANY PROFILE
(PF) PERSONNEL FILES
(PP) PAYROLL PROCESSING
(RG) REPORT GENERATOR
(EP) EXIT 'PAYDAY' Program
```

For the initial setup, the "Company Profile" option must be chosen so that vital information about your company can be recorded, such as name, location, pay period, FICA withholding percentage and the names of individual departments within the company. The program allows for up to six different departments. Also, there is an option that allows you to configure the system for flexible disks or a hard disk.

Any or all of the initial information can be edited, including the FICA percentage withholding value, so any alterations, even after the data has been recorded, can be changed without destroying previous payroll information. In fact, any information can be changed and rechanged during a single session with the system.

Once the Company Profile information is completed, a list of employees must be entered via the "Personnel Files" option, PF, from Central Control. The Personnel Files menu lists seven selections:

```
(AN) ADD NEW employee's file to
     our records
(LT) LEAF THROUGH existing
     employee files
(PC) Print PAPER COPY of all or
     some employee files
(ML) Print employee MAILING
     LABELS
(SN) SEARCH for a file by an
     employee's last NAME
(SF) SEARCH for a file by an
     employee's FILE #
(CC) Return to CENTRAL CONTROL
```

After AN is chosen, the program asks for information about the employee: name, address, telephone, Social Security Number, pay-period wage, FICA withholding, federal withholding, state withholding, withholding status (single, married and the number of dependents), department, and control number. All of the information must be entered except for the control number. This is an optional entry depending on whether your company assigns identification codes to employees.

After all the employee information is entered, the Personnel Files menu is redisplayed. From the menu, the program allows various options for

working with the employee roster. Changes in any or all employee information can also be made, including the employee's current status, active or inactive. Besides having a detailed database, the wage and tax information is used by the program to automatically compute each individual's pay during the "Payroll Processing" option of Central Control.

The Payroll Processing section of *Payday* is easy to use. After PP is entered from Central Control, the "Payroll Processing Active Employee Roster" is displayed, listing all the active employees. Each employee is numbered according to the order in which they were originally entered. Below the roster are five options for processing the payroll:

```
(#) Begin accepting this
    employee's current payroll
    data
(#C) CHANGE Active/Inactive
     status of this employee
(BP) BEGIN PRINTOUT - All data
     for this pay period has
     been entered
(NP) Turn to NEXT PAGE for addi-
     tional employee listings
(CC) No payroll has been entered
     - return to CENTRAL CONTROL
```

Each employee's payroll must be computed individually. This is done by going through the employee list, one by one, and entering the number associated with each person. After the desired number is entered, a summary screen appears giving all the information concerning that employee. At this point any or all the information about the employee can be altered before his pay period is entered. You may also choose to enter the pay information manually or have it automatically computed from the personnel file data.

When everything is entered, the screen is cleared and the program asks for the hours worked for the pay period if the employee is on an hourly rate. Overtime and the overtime rate and special deductions are entered, if any. Once completed, the program automatically computes the payroll for that employee, including FICA, federal, and state withholding taxes. A time-saving feature of the program is its built-in tax tables, both federal and state. These tables are inaccessible through the program, but if changes do occur in the tax regulations and rates (state or federal), Heath Research will provide a tax table update diskette for a fee of $12.

After each employee's payroll is computed, the information is recorded and the Payroll Processing menu is again displayed, allowing you to choose another employee. As each employee's pay period is completed, the employee's name is deleted from the roster — a helpful feature if one is working with a large employee list. When the entire payroll is completed, a printout must be obtained in order to see the computed pay information: pay period, gross wages, withheld taxes, miscellaneous deductions, and net wages. If any of the information is incorrect, you must reprocess the payroll.

The last option of the Central Control menu is the "Report Generator." Its menu allows six different reports to be printed:

```
(PP) Individual PAY PERIOD
     Report
(CS) COMPREHENSIVE Payroll
     STATEMENT
(MR) MONTHLY Payroll REPORT
(QR) QUARTERLY Payroll REPORT
(YR) YEARLY Payroll REPORT
(W2) End-of-year W2 forms
```

Each of these reports is invaluable in keeping track of your payroll. In particular, the W2 report is helpful in that it will actually fill out the W2 forms

on the printer, using blank W2 forms on continuous-form computer paper.

The "Comprehensive Payroll Statement" is also a good managerial tool. It is a hardcopy of the company payroll broken down department by department. In addition, a chart is printed comparing the gross payroll of each department within the company and the year-to-date totals of each department to the previous year's totals. These reports are generated automatically and are the best feature of *Payday*.

Since the program does not use any special printer graphics, almost any 80-column printer can be used for the reports. Also, it isn't necessary to have a graphics board installed; the screen displays are presented in standard characters.

The instruction manual, a three-ring loose-leaf binder, is concise and guides you easily through the operation and functions of the program. At the back of the binder are tabbed sections where the printed payroll reports can be stored – a convenient feature.

The program does, however, have a few pitfalls. One missing feature is the lack of screen displays of the reports. Having to print out the entire payroll to check for errors is a serious drawback. It would have been more convenient to view the entire payroll on the screen before getting a hardcopy. Also, the program can only accommodate one miscellaneous deduction item, which limits the applicability of the program to firms with retirement, savings, or other programs that require more than one deduction item.

Despite these disadvantages, *Payday* is a fast, easy-to-use system that should fill the payroll needs of your small business.

(Heath Research, 3841 Croonenbergh Way, Virginia Beach, VA, 23452, $125)

— Ralph Rideout

Software **1000/2000**

PC Privacy: Personal and Confidential

In these days of whiz-bang multifunction programs that try to be everything to all people, it is refreshing to find a program which does one thing and does it well. *PC Privacy* by MCTel, Inc. protects the sensitive information you have painstakingly stored on computer media. It does its job thoroughly and quickly, and it couldn't be easier to use.

If you're like most computer users, you are probably unaware of how easy it is for anyone with even the most rudimentary knowledge of MS-DOS to gain access to the information stored in your computer's files. If you keep files such as employee records, financial projections, or even a database file containing names, addresses and telephone numbers of clients, prospects and others you do business with, a competitor or malicious employee can usually discover more than they should know with a simple TYPE, PRINT or DEBUG command.

If your files are all on floppy disks, you can gain some measure of security by locking them in a safe or otherwise restricting access to them, but if you use a hard disk, you are particularly vulnerable because your files are always in the computer. Even if you delete a file using ERASE, the information in that file is not actually deleted until you overwrite it by saving a new file. I once worked for a firm whose hard-disk PC went out for repair and was temporarily replaced by a loaner. To everyone's surprise, the loaner arrived loaded with the previous owner's programs and datafiles. What no one realized at the time was that our hard disk could just as easily find its way into someone else's computer.

PC Privacy helps to solve the dilemma of file security by thoroughly scrambling your files and making them undecipherable to anyone who doesn't have the proper "key." The program disk contains three small programs: *Encrypt, Decrypt* and *Purge*. Each of them is straightforward and easy to use. To encrypt a file, you simply type ENCRYPT, the name of the file you want to encrypt and a phrase which acts as the key. To decrypt the file, you simply type DECRYPT, the name of the file you want to decrypt and the same phrase which was used to encrypt it. Any other key will fail to unscramble the file.

The entire process is quick, simple, and very effective. You must, of course, be careful not to forget the key used to encrypt the file, or you will have no

way of decrypting it. The short manual which comes with the program warns that not even MCTel can decrypt the file. I don't know that it is absolutely impossible to crack the encryption process, and I would hesitate to entrust any national security secrets to the program, but for everyday use it certainly seems effective enough.

I timed the programs on my 2000 with a hard disk. The smallest file (4 bytes) was encrypted and decrypted almost instantaneously, in under two seconds. A 640,000-byte file required about five minutes, and a medium-sized file of 75K took only 40 seconds. The encrypted files were generally about one-third larger than the originals, with the length of the key having no apparent effect on encrypted file size.

The *Encrypt* program will not allow you to enter the same filename for the encrypted file as the original, preventing you from overwriting it by mistake. *Decrypt*, on the other hand, will allow you to overwrite an existing file, but warns you before doing so, and gives you a chance to change your mind. Neither program will accept illegal filenames. The third program, *Purge*, allows you to completely erase an encrypted file, so no trace of it remains on your disk.

Two other points show the care which went into designing *PC Privacy*. First, after you type in the key used for encrypting, the program scrolls the screen up, preventing anyone walking by from seeing the key while you are encrypting a large file. Also, the encrypted file does not contain any unprintable control characters. If you transmit files by modem regularly, you are probably aware that many programs store their data in a way that prevents them from being transmitted unless you use a special protocol. Files encrypted with *PC Privacy* need no special handling and may be transmitted as straight ASCII files. Thus, the receiving end needs only its own copy of *PC Privacy* (and, of course, the key) to be able to use your *Wordstar, Lotus 1-2-3* or other files.

If the security of your information is important to you, I recommend you evaluate *PC Privacy* for your needs.

(MCTel, Three Bala Plaza East, Suite 505, Bala Cynwyd, PA 19004, 215-668-0983, $140)

— V. Scheluchin

Software 1000/1200

Magic Menu: A Time-Saving Program Manager

It seems like every office or home with a computer also has a person to whom everyone goes with questions: "Tell me just once more: how do I load the word processor?"; "But how do I get the printer into correspondence mode?"; "What's a directory?" If you are that person you might want to consider *Magic Menu* by DeereSoft.

Using *Magic Menu*, the person who is tired of coming up with all of the answers can create a system of menus which will allow the more casual user to select and execute any of over 1,000 programs without having to deal with the dreaded DOS or open a manual.

Hardware requirements call for an IBM PC or compatible with two dual-sided floppy drives or one dual-sided floppy and one fixed drive, 128K, and PC or MS-DOS 2.0 or higher. The manual indicates that the program is also supported on the Tandy 1000 and 1200HD, and is 98% compatible with the Tandy 2000.

The program comes on a single-sided diskette which is not copy protected. It actually consists of two programs: *Magic Menu* and *Magic Menu Editor*. While *Magic Menu* insulates the user from DOS, the editor which creates the menus requires a firm grasp of the operating system. This is compounded by the fact that the documentation, while well-written and attractive in its three-ring format, is somewhat brief. A text file on the diskette expands upon the printed documentation, and should have been incorporated in the manual. Using both of these sources, I still found considerable experimentation necessary to fathom some aspects of the editor.

DeereSoft indicates that *Version 2.0*, which is now being shipped, includes expanded documentation which may well eliminate this criticism. If you are willing to spend some time to master the editor, you should find *Magic Menu* a powerful system.

The display is designed for use with

a color monitor, and foreground, background and border colors can be selected. My Tandy 1200HD with monochrome monitor and color/mono graphics board produced a satisfactory display without modification in all respects but one: data entry. The program interrogates the system, and, finding a graphics board, assumes a color monitor. The foreground/background combination used for data entry produces no display on a monochrome monitor. The addendum briefly suggests modifying a configuration file on the disk to override the offending color combination, but I had no better results after the suggested modification. Deere-Soft indicates that this problem has also been solved in *Magic Menu 2.0*.

To get the most out of *Magic Menu*, the AUTOEXEC.BAT file should be modified to route the user directly into the menu system. Following power-up, the first menu screen (created by the editor) appears. Selection of an entry from a menu screen will have one of several designated results. Selection from a menu can result in the display of a subsequent menu. For example, the selection of "Word Processing" from a menu screen could result in the display of a menu with several word processing programs to choose from. Selection can also result in the execution of a program or batch file, preceded by a change in directory if necessary. When the program called from the menu system is ended, the user is returned to the menu system. The result of a selection can be the display of a message or prompt, the output of a character string to a device (to change the printer to correspondence mode, for example), or the execution of a DOS command. The last result available is to define a system-wide variable, much the way parameter passing is used within DOS batch files.

Another type of menu screen available is sequence screen. Rather than displaying choices, this screen executes a series of entries automatically. Thus, the menu screen for a particular word processor could include selections for draft or correspondence modes. Choosing one of these would activate a sequence screen which would send a character string to the printer to set the right mode, and then execute the word processing program.

Password protection can be placed on any selection on any screen, and can also be applied to the return-to-DOS function F10. Since the user may CTRL ALT DEL to DOS and use the *Magic Menu Editor* to view (or change or delete) the passwords, this protection will only stop inexperienced users. *Version 2.0* encrypts passwords, but this still cannot prevent the alternate method of returning to DOS.

After using the editor to modify the menu system, the menu definition file must be copied from the MAGICM directory to the root directory where it will be used by *Magic Menu*. One must return to DOS to accomplish this, or create a menu selection to handle the copy routine as I did, but it would be more convenient to be able to perform this task from the editor.

Once the complexities of the editor have been mastered and a menu system has been created, *Magic Menu* is a breeze. Anyone should be able to choose their way through the menu screens and execute the program they want without coming to you with "How do I . . .?" The more experienced user can skip directly to the desired screen and selection without following the

Complete Accounting Software
Including Payroll for $15 per module

101 Profitable Reasons To Buy MCBS Accounting Software!

General Features

1) Menu Driven System
2) Integrated or Stand Alone
3) Simplified Installation
4) Complete Audit Trails
5) Data Files can be accessed by Lotus, dBase, etc.*
6) Manuals on diskettes, print them on your printer.
7) Modules can be purchased separately.
8) Accounts and transactions limited only by disk space.

General Ledger

9) Supports multiple departments
10) Flexible chart of accounts.
11) Prints financial statements in flexible, user defined formats.
12- Prints Income Statement and
17) Balance Sheet by current month, current quarter, and 3 previous quarters.
18- Prints ***Departmental*** Income
23) Statement and Balance Sheet by current month, current quarter, and 3 previous quarters.
24- Prints ***Budget Comparison***
25) Income Statement and Balance Sheet comparing Actual vs. Budget with variances and ratios.
26- Prints Trial Balance and Chart
27) of Accounts
28) Prior Period Adjusting Entries do not impact current results.
29) Data Entry screen displays beginning, current and ending balances as well as the last 10 entries.
30) System allows new accounts to be created during data entry.

Accounts Payable

31) Provides for entry, updating and deletion of vendor information.
32- Allows entry, editing and posting
33) of payables, adjustments, cancellations and prepaids.
34- Prints the Open Item & Cash
35) Requirements reports.
36) Allows partial payments.
37) Prints A/P checks
38) Prints Check register.
39) Prints the A/P distributions to the General Ledger
40) Prints the vendor anaylsis report.
41) Interfaces to MCBS' General Ledger or may stand alone.

Accounts Receivable

42) Provides for entry, updating and deletion of customer infomation
43) Allows entry, editing and posting of receivables, credit memos, cash payments.
44) Allows up to 99 repetitive line items for fast invoicing.
45) Prints the open item and aged receivables reports.
46- Prints Invoices, Credit Memos
48) and Statements.
49) Prints Sales Analysis.
50) Supports Partial Payments
51) Prints Invoice, Payment and Credit Memo Registers.
52) Interfaces to MCBS' General Ledger or may stand alone.

Payroll
with State & Local Taxes

53) Provides for entry, updating and deletion of employee information.
54- Handles hourly, salaried or
56) commissioned employees on weekly, biweekly, semi-monthly or mothly pay periods.
57) Calculates Federal, State and Local Taxes.
58) Allows user to maintain all tax tables.
59) Maintains payroll history.
60) Calculates the payroll.
61) Prints the payroll register, check register and deductions register.
62) Prints payroll checks.
63) Provides for after-the-fact payroll
64- Provides W-2's, 1099's , and 941's
67) & prints quarterly earnings reports.

Inventory
with Order Entry & Purchase Orders

68) Uses for LIFO and Standard Cost.
69) Provides for Inventory item entry, updating and deletion.
70) Provides for order entry and editing with inventory update during entry.
71) Stock and non-stock items.
72) Handles purchases, returns, shipments and adjustments.
73) Prints invoices, credit memos, picking tickets, purchase orders, price lists and shipping labels.
74) Prints on preprinted forms or plain paper.
75) Prints stock status report and purchasing advice reprot.
76) Full back order retention.
77) Quarterly, yearly and prior year inventory history.
78) Performs book to physical analysis.
79- Over 20 management reports
99) including extensive Sales Analysis and Lost Sales Report.
100- Interfaces to MCBS' General
101) Ledger, A/R, A/P or may stand alone.

Any Module For $15 ea.
Or All 5 For $50 Total

TO ORDER CALL:
1-800-334-0854, Ext. 826
For More Information:
call (713) 444-6269

Accountants ONLY ...

Accountant's Client Write-up System

1) Designed for Accountants
2) Allows multiple clients on same disk, accessed by client code.
3) Allows new accounts or new employees to be entered from data entry screen.
4) Supports multiple departments.
5) Running balance shown on screen during data entry.
6) Payroll entries made from same entry screen as other entries.
7) Accumulates all payroll information and prints Quarterly payroll reports including 941's
8) Prints W-2's and 1099's.
9) Produces client bill on a variety of bases.
10) Produces more than 12 reports including Changes in Financial Position, Income Statement, Balance Sheet, Detailed General Ledger.
11) Includes automatic report printing.

$150

All programs run on IBM (PCjr/PC/XT/AT), AT&T, EPSON, TANDY (1000/1200/2000), TI, WANG, COMPAQ, CORONA, SANYO, COLUMBIA*, and MS-DOS machines.

Requires 128K Ram, 2 Floppies or Fixed Disk, 80 Column Printer, MS-DOS, PC-DOS 2.0 or later.

**Computer & software names are trademark or registered by their respective manufacturers.*

MONEY BACK GUARANTEE!

Service Contract
Telephone consultation - $50/yr.
Shipping charges $5.00
Texas Resident add 6⅛% Tax
COD Charges $5.00
Payment by Check, MC or VISA

ATTENTION DEALERS!!!! Call or Write Now for the VERY BEST MARGINS (over 50%) in the Industry.

MCBS **Micro Computer Business Services**
12703 Stuebner Airline Houston, TX 77014

Complete Package G L A P A R P R INV QTY ____ X $ 50 00 ea = ____
Individual Module - G L A P A R P R INV QTY ____ X $ 15 00 ea ____
Accountants' Client Write-up System QTY ____ X $150 00 ea - ____
Pay Method Prepaid, Check Enclosed ____ Texas Residents Add 6⅛% = ____
MC. #____ Shipping Charges = 5.00
VISA. #____ If COD add $5 00 ____
Expires ____ Signature ____ TOTAL ____

Company Name ____ Contact ____
Street Number ____ Phone ____
City ____ State ____ Zip ____

route through the menu system.

If you have many programs to deal with, and don't want to have to remember (or tell someone else) the filespecs to execute them, or the steps necessary to set up the peripherals, or if you just want to simplify your day-to-day dealings with DOS, look into *Magic Menu*. In spite of a few shortcomings, some of which have apparently been addressed in *Version 2.0*, I would recommend this program management software.

(DeereSoft, Inc., P.O. Box 1360, Melbourne, FL 32901, $99.00)

— Stanley Townsend

Software **1000/1200/2000**

Getting Trim with *Diet Wise*

For those of us who have been sitting at the computer for hours at a time, adding inches around the middle, *Diet Wise* software might be a practical purchase.

The program doesn't display nasty warning messages when you've exceeded 3,000 calories for the day, does not refuse to allow pizza or another beer in the diet, nor does it insist that you eat six grapefruit or ten bananas a day. Instead it counts the calories consumed in a day and calculates daily requirements of fifteen vitamins and minerals.

Diet Wise version 3.05 runs in monochrome on the Tandy 1000, 1200 or 2000. Reports can be printed on the screen or output to an optional printer.

The program has three separate functions: a data input file with up to 50 different foods that produces a daily goal graph; a nutrient file which compares food values of up to twenty items; and a recipe/menu calculator that breaks down calories and vitamins in a recipe into individual calculations.

Diet Wise has a database of food values of 800 foods. Another 300 items can be added to the database.

The "nutrient goal graph" shows recommended daily allowances for vitamins and minerals, as well as Heart Association guidelines and U.S. dietary goals. The graph is personalized by an individual's age, sex, weight and nutrient intake. A personalized chart for each member of the family can be kept.

Entering data is very simple. Breakfast is input with: "1 cu orange juice, 1 lg egg, 1 sl white bread, and 1 cu coffee." After the food eaten during a day is input, a listing appears on screen which can be edited. (The edit function even allows the 4 oz. martini to be cut back to a 2 oz. one!)

A goal graph profile is the next step. Age, sex, height, current weight and desired weight are prompted, and the demographic information appears for verification. If you add five inches to your height to compensate for the 200 pounds, the computer will believe you.

This goal graph is a nifty feature of the program. A bar chart is displayed showing the percentage of calories, sugar, alcohol, fat, carbohydrates, sodium, cholesterol and vitamins consumed in a day. If you've exceeded the recommended allowances, the program shows an "eat less" prompt. If you've failed to consume enough iron or vitamins, the program gently nags "eat more!"

I was fascinated with the analysis of a typical day's diet. I included my megavitamin tablet in the database, and added my food intake to the program. Even though I ate a balanced diet, and took my daily pill, I was low in vitamin B-2 and iron.

The high-low scan table in the program is very useful. In order to cut back on calories or increase consumption of calcium, a chart can be listed for up to 4 food groups. Foods consumed during the day are ranked from highest to lowest, so it is easy to determine which foods high in calories to avoid, or the ones highest in minerals to include in the diet.

Perhaps you want to compare the nutrients in an apple with those in a peach. The database shows the fifteen nutrients in each fruit, and you can compare the calories, the amount of sodium or of vitamin B-1.

The recipe calculator portion of the program is helpful. All the ingredients in a recipe are input, and the number of servings is entered. The recipe file is saved to disk by name, so "Ken's Meatloaf" is included in the database, and an entry of "1 sv Ken's Meatloaf" automatically calculates the nutrients in each individual portion.

The label function allows addition of foods to the database with information obtained from the food label. Perhaps you eat a bowl of "Crunchy Pruneflakes" cereal every morning, and want to calculate how much nutrition is in each serving. Once the data from the side of the box is entered into the *Diet Wise* program, you can call up the values of a bowl of this cereal.

A very useful section of the operating manual is an excellent description of fifteen nutrients. Included are calories, protein, carbohydrates, sugar, fat, alcohol, vitamin A, vitamin C, thiamin (B-1), riboflavin (B-2), niacin, calcium, iron, cholesterol and sodium. The updated version of this software will also include potassium levels.

The information section shows the measurement factor (grams, milligrams or international units) and the recommended daily level for adults and children for each nutrient. Reasons why each is useful are explained, and good food sources for each vitamin or mineral are listed.

The *Professional Nutrient* software program includes a few enhancements to the basic package. The expert mode is streamlined; 900 foods are included in the database, plus the 300 personal favorites which can be added.

A chart compares three days rather than each day. Averages for the three days are shown. An activity log determines the energy expenditure for exercises such as running, washing dishes or debugging a program at the computer. The results of the activity log are interfaced with the calorie count in the graph program.

The professional package also has a database editor. One feature lacking in the personal package is the ability to change food values once they have been entered. If you showed a hot fudge sundae at 50 calories by mistake, the $120 package would not allow editing.

The professional package also allows the user to enter four more nutrients to the database. A price function may be added to the recipe calculation, and a price per serving is automatically calculated.

(*Diet Wise*, $120.00, or the *Nutripak Professional System*, $349, are available from Nutritional Data Resources, P.O. Box 540, Willoughby, OH 44094, 216-951-6593.)

— M.J. Batham

The following products recently have been received by PCM, examined by our magazine staff and approved for the PCM *Seal of Certification*, your assurance that we have seen the product and have ascertained that it is what it purports to be. This month the *Seal of Certification* has been issued to:

Tandy 2000 DeskMate, an integrated software package including a text editor, work sheet, telecommunications, electronic mail and calendar. Similar to the program included with all Tandy 1000s. *Tandy, distributed through Radio Shack stores nationwide. Requires a 256K Tandy 2000, Catalog No. 26-5278, $199.95.*

DeskSet, provides "desk-top" tools, including calculator, note pad, clip board, "Pop-DOS," calendar, alarm clock and a telephone dialer. Tools can be accessed while running other applications and data can be transferred between tools and application. Adds about 128K to memory requirement of application running. *Bellsoft, 2820 Northup Way, Bellevue, WA 98004. Requires Tandy 1000 or 1200, $69.95.*

DR Graph, a software package allowing the user to draw graphs and charts on the display and plot or print the image. *A Digital Research product distributed through Radio Shack stores nationwide, Tandy 2000, Catalog No. 26-5278, $195.00.*

HomeBase, a desk-top organizer program offering telecommunications, mailing label printing, a free-form database with cross-indexing and background electronic mail, in addition to features found on most desk-top programs. *Amber Systems, 1711 S. Saratoga-Sunnyvale Road, San Jose, CA 95129. Requires Tandy 1000 or 1200, $49.95.*

Learing BASIC for the Tandy 1000/2000, a book providing a step-by-step course in BASIC for both the Tandy 1000 and 2000. Short, precise chapters with exercies that move you along through each subject. Chapters cover strings, math, display formatting, arrays, program control and sound. *Tandy, available through Radio Shack stores nationwide, Catalog No. 25-1500, $19.95.*

Macro*Track, an economic forecasting system for use by any manager or investor. Displays graphs and charts without technical jargon. *Black River Software Corporation, 118 North Marshall Street, Suite 150, Winston-Salem, NC 27101. Requires Tandy 1000 or 1200, $299.95.*

MS-DOS Volume 1 — The Basics, a book by David A. Lien and Gary Williams, is the first of a two-volume set on the MS-DOS operating system. Commands are broken down into simple, straightforward instructions. No attempt is made to go into technical detail in this volume. The book deals specifically with Tandy's MS-DOS computers. *Tandy, available through Radio Shack stores nationwide, Catalog No. 25-1506, $7.95.*

PowerText Formatter, a word processing "add on" designed to work with *pfs:Write*, the *IBM Writing Assistant* and other word processors. The program enables users to produce documents with justified text, true proportional text, footnotes, multiple columns, outlines and other complex styles. *High Performance Computer Products, 417 Halstead Avenue, Harrison, NY 10528. Available for Tandy 1000 and 1200, $49.95.*

Remote Disk, an easy-to-use file storage program allowing Model 100 owners to connect their portable to a Tandy desk-top computer and save and retrieve files on the larger computer's floppy or hard disk. Requires 16K Model 100 and one of the following Tandy computers: Models I, II, III, 4, 4P, 12, 16, 2000, 6000 or Color Computer. *Tandy, distributed through Radio Shack stores nationwide, Catalog No. 26-3839, $59.95.*

Dac-Easy Accounting, a program for the Tandy 1000, 1200 HD or 2000, is seven integrated modules on one disk including General Ledger, Accounts Receivable, Accounts Payable, Purchase Order, Billing, Inventory and Forecasting. With Forecasting users can build a three year history of their business to plan the coming year. *DAC Software, 5580 Petersen, Suite 130, Dallas, TX 75240, $69.95 plus $7.50 S/H, direct order phone (214) 458-0038.*

The Tandy 200 Portable Computer, a learner's manual by David A. Lien. The book covers word processing, telecommunications, schedule keeping, address directory, spreadsheet and a tutorial on BASIC. Written in the highly-successful tutorial style that has become Lien's trademark. *Tandy, available through Radio Shack stores nationwide, Catalog No. 26-3869, $19.95.*

TMPC, a program designed to help manage "your most precious commodity" — time. Features include a perpetual calendar, deadline capability and "to do" tracking. Requires a 16K Model 100. *Acroatix Incorporated, P.O. Box 273, Wilmington, MA 01887, cassette, $49.95.*

WriteROM, a Model 100 word processing program that provides features such as centering, setting margins, justification and headers and footers. The program is contained on a snap-in ROM that plugs into the bottom of any Model 100. *Portable Computer Support Group, 11035 Harry Hines Boulevard #207, Dallas, TX 75229. Available for the Model 100, $149.95.*

By awarding a *Seal*, the magazine certifies the program *does exist*, but this *does not* constitute any guarantee of satisfaction. As soon as possible, these hardware or software items will be forwarded to PCM's reviewers for evaluation.

Using *BAREAD 2.1*

Bar code listings must be read in numerical order beginning with Line 1 and continuing through the last line of the listing. The computer display is used to prompt you as to which line to scan and give you warning messages should you happen to get out of step.

When you run *BAREAD*, it asks you to scan the first line of the bar code listing. This line contains the name of the program as well as the beginning of the program itself. The computer will sound a high-pitched beep whenever it's ready for you to scan a line. After a line has been successfully read, you'll hear a lower beep. A "blip-bloop" sound prompts you to turn your attention to the screen for a message. You'll hear this when you accidentally scan a line out of sequence.

After reading the first line, you continue scanning with the second line. Remember to wait for a high beep before scanning and then listen for a low beep to indicate a successful read.

Once the last line of the listing has been scanned, *BAREAD* will return control to the Tandy 100/200 menu screen. Note that the program you just scanned is now in the directory with a .DO extension.

The final step is to convert the .DO text file to a normal BASIC program. This is done quite simply by going to BASIC and loading the file with a command such as LOAD"TEST .DO" (if the program name were TEST). The program will load into BASIC and will be ready to run. To save the program in BASIC's compressed format (.BA extension), you'd type SAVE"TEST" (if the program were named TEST). You may then kill the .DO file with KILL "TEST.DO".

BAREAD 2.1

```
1ØØØ ' *** Initialize ***
1Ø1Ø ON ERROR GOTO 1Ø4Ø
1Ø2Ø CLEAR 1ØØØ:MAXFILES=2
1Ø3Ø GOTO 1Ø5Ø
1Ø4Ø IF ERR=5 THEN RESUME NEXT
1Ø5Ø ON ERROR GOTO Ø
1Ø6Ø RUNM "B3OF9"
1Ø7Ø OPEN "WAND:" FOR INPUT AS #1
1Ø8Ø UC%=-1
1Ø9Ø PC$="Ø123456789ABCDEFGHIJKLMNOPQRST
UVWXYZabcdefghijklmnopqrstuvwxyz- $+"
11ØØ DIM RW$(36)
111Ø ER$(1)="You must scan line 1 first!
"
112Ø ER$(2)="You've SKIPPED a line!"
113Ø ER$(3)="You've ALREADY SCANNED this
 line!"
114Ø ER$(4)="Code not PCM2/39 format!"
115Ø ER$(5)="Command not applicable here
!"
116Ø ER$(6)="You cannot skip this line!"
117Ø ER$(7)="Selected resume file not in
 computer!"
118Ø ' *** Read Reserved Words List ***
119Ø DATA BEEP,CLEAR,CLOSE,DATA,DEFDBL,D
EFINT,DEFNG,DEFSTR,ELSE,GOSUB,GOTO
12ØØ DATA INKEY$,INPUT,INSTR(,LCOPY,LEFT
$(,LINE(,LOADM,LPRINT,USING,MAXFILES
121Ø DATA MID$(,NEXT,PEEK,POKE,POWER,PRE
SET(,PRINT,READ,RESTORE,RETURN,RIGHT$(
122Ø DATA SOUND,SPACE$(,STRING$(,THEN
123Ø FOR I%=1 TO 36:READ RW$(I%):NEXT I%
124Ø ' *** Procedure Begins Here ***
125Ø CLS:PRINT@44,"PCM Bar Code Program
Reader v2.1"
126Ø LINE(2Ø,4)-(219,18),1,B:LINE(22,6)-
(217,16),1,B
127Ø NN%=1
128Ø GOSUB 166Ø:IF ER%>Ø THEN GOSUB 162Ø
:GOTO 128Ø
129Ø IF LL%=Ø AND INSTR("YN",IL$)>Ø THEN
 ER%=5:GOSUB 162Ø:GOTO 128Ø
13ØØ IF LL%=Ø THEN ON INSTR("ALSR",IL$)
GOTO 182Ø,189Ø,198Ø,2Ø5Ø
131Ø IF LL%=1295 THEN 135Ø
132Ø IF LL%<>NN% AND NN%=1 THEN ER%=1:GO
SUB 162Ø:GOTO 128Ø
133Ø IF LL%<NN% THEN ER%=3:GOSUB 162Ø:GO
TO 128Ø
134Ø IF LL%>NN% AND NN%>1 THEN ER%=2:GOS
UB 162Ø:GOTO 128Ø
135Ø IL$=RIGHT$(IL$,19)
136Ø IF LL%=1 AND NN%>Ø THEN GOSUB 178Ø
137Ø CL$=CL$+IL$
138Ø FOR I%=1 TO LEN(CL$)
139Ø    CH$=MID$(CL$,I%,1)
14ØØ    IF CH$="%" THEN GOSUB 151Ø:IF NL
% THEN 147Ø ELSE GOTO 144Ø
141Ø    IF CH$="/" THEN GOSUB 155Ø:IF NL
% THEN 147Ø ELSE GOTO 144Ø
142Ø    IF CH$="." THEN UC%=NOT(UC%):GOT
O 145Ø
143Ø    IF CH$=>"A" AND CH$<="Z" AND NOT
(UC%) THEN CH$=CHR$(ASC(CH$)+32)
144Ø    XX$=XX$+CH$:IF RIGHT$(XX$,1)=CHR
$(13) THEN PRINT#2,XX$;:XX$="":UC%=-1
145Ø NEXT I%
146Ø CL$=""
147Ø PRINT@2ØØ,SPACE$(8Ø);
148Ø IF LL%<>1295 THEN NN%=LL%+1:GOTO 12
8Ø
149Ø ' *** Done ***
15ØØ CLOSE:CALL 618Ø7!:CLEAR 5ØØ,HIMEM:M
```

Submitting Material to PCM

Contributions to PCM are welcome from everyone. We like to run a variety of programs which will be useful/helpful/fun for other Tandy Portable and MS-DOS computer owners. We now support the Model 100, the Tandy 200, and the Tandy models 2000, 1200 and 1000.

Program submissions must be on tape or disk, and it is best to make several saves, at least one of them in ASCII format. We're sorry, but we do not have time to key in programs. All programs should be supported by some editorial commentary explaining how the program works. Generally, we're much more interested in how your submission works and runs than how you developed it. Programs should be learning experiences.

Pay for submissions is based on a number of criteria. The rate of remuneration will be established and agreed upon prior to publication.

For the benefit of those who wish more detailed information on making submissions, please send a SASE to: Submissions Editor, PCM, P.O. Box 385, Prospect, KY 40059. We will send you comprehensive guidelines.

Please do not submit programs or articles currently submitted to another publication.

If you feel qualified to review software and or hardware products for computers covered in PCM, send us your name, address and phone number; we will send you a questionnaire form and a copy of our reviewer guidelines.

```
ENU
151Ø ' *** Decode Reserved Word ***
152Ø NL%=Ø:IF I%>LEN(CL$)-1 THEN NL%=-1:
CL$="%":GOTO 154Ø
153Ø I%=I%+1:CH$=RW$(INSTR(PC$,MID$(CL$,
I%,1)))
154Ø RETURN
155Ø ' *** Decode Hex and Control Charac
ters ***
156Ø NL%=Ø:IF I%>LEN(CL$)-1 THEN NL%=-1:
CL$="/":GOTO 161Ø
157Ø I%=I%+1:IF INSTR("/%.",MID$(CL$,I%,
1))>Ø THEN CH$=MID$(CL$,I%,1):GOTO 161Ø
158Ø IF I%>LEN(CL$)-1 THEN NL%=-1:CL$=RI
GHT$(CL$,2):GOTO 161Ø
159Ø HX$=MID$(CL$,I%,2):CH$=CHR$((INSTR(
"Ø123456789ABCDEF",LEFT$(HX$,1))-1)*16+I
NSTR("Ø123456789ABCDEF",RIGHT$(HX$,1))-1
)
16ØØ I%=I%+1
161Ø RETURN
162Ø ' *** Error Codes ***
163Ø SOUND 5ØØØ,1Ø:SOUND 8ØØØ,1Ø:SOUND 5
ØØØ,1Ø
164Ø PRINT@22Ø-.5*LEN(ER$(ER%)),ER$(ER%)
;
165Ø RETURN
166Ø ' *** Get Code Line ***
167Ø PRINT@173,"";:PRINT USING "Scan lin
e ###";NN%
168Ø IF NN%=-1 THEN PRINT@173,"Scan any
line":GOTO 17ØØ
169Ø SOUND 5ØØ,5
17ØØ INPUT#1,IL$:ER%=Ø
171Ø FOR I%=1 TO LEN(IL$)
172Ø IF MID$(IL$,I%,1)="!" THEN MID$(IL$
,I%,1)="."
173Ø NEXT I%
174Ø IF LEN(IL$)<>1 AND LEN(IL$)<>21 THE
N ER%=4:RETURN
175Ø IF LEN(IL$)=1 THEN LL%=Ø:RETURN
176Ø LL$=LEFT$(IL$,2):LL%=(INSTR("Ø12345
6789ABCDEFGHIJKLMNOPQRSTUVWXYZ",LEFT$(LL
$,1))-1)*36+INSTR("Ø123456789ABCDEFGHIJK
LMNOPQRSTUVWXYZ",RIGHT$(LL$,1))-1
177Ø RETURN
178Ø ' *** Open Program File ***
179Ø PN$=LEFT$(IL$,6):IL$=RIGHT$(IL$,LEN
(IL$)-6)
18ØØ OPEN PN$ FOR OUTPUT AS #2
181Ø RETURN
182Ø ' *** Abort ***
183Ø BEEP:BEEP:BEEP
184Ø PRINT@2Ø9,"ABORT!  Are you sure?";
185Ø INPUT#1,AN$
186Ø IF INSTR("YN",AN$)=Ø THEN BEEP:PRIN
T@251,"Scan 'YES' or 'NO'":GOTO 185Ø
187Ø PRINT@2ØØ,SPACE$(8Ø);
188Ø IF AN$="Y" THEN CLOSE:KILL PN$+".DO
":GOTO 149Ø ELSE GOTO 128Ø
189Ø ' *** Skip Line ***
19ØØ IF NN%=1 THEN ER%=6:GOSUB 162Ø:GOTO
 128Ø
191Ø BEEP:BEEP:BEEP
192Ø PRINT@21Ø,"SKIP!  Are you sure?"
193Ø INPUT#1,AN$
194Ø IF INSTR("YN",AN$)=Ø THEN BEEP:PRIN
T@251,"Scan 'YES' or 'NO'":GOTO 193Ø
195Ø PRINT@2ØØ,SPACE$(8Ø);
196Ø IF AN$="Y" THEN NN%=NN%+1
197Ø GOTO 128Ø
198Ø ' *** Stop & Save ***
199Ø BEEP:BEEP:BEEP
2ØØØ PRINT@2Ø7,"STOP & SAVE! Are you sur
e?";
2Ø1Ø INPUT#1,AN$
2Ø2Ø IF INSTR("YN",AN$)=Ø THEN BEEP:PRIN
T@251,"Scan 'YES' or 'NO'":GOTO 2Ø1Ø
2Ø3Ø PRINT@2ØØ,SPACE$(8Ø);
2Ø4Ø IF AN$="Y" THEN 149Ø ELSE GOTO 128Ø
2Ø5Ø ' *** Resume ***
2Ø6Ø IF NN%<>1 THEN ER%=5:GOSUB 162Ø:GOT
O 128Ø
2Ø7Ø PRINT@254,"Resume Mode";
2Ø8Ø NN%=1:GOSUB 166Ø
2Ø9Ø IF LL%=Ø THEN ER%=5 ELSE IF LL%<>1
THEN ER%=1
21ØØ IF ER%>Ø THEN GOSUB 162Ø:GOTO 2Ø6Ø
211Ø PN$=MID$(IL$,3,6)
212Ø ON ERROR GOTO 214Ø
213Ø OPEN PN$ FOR INPUT AS #2:GOTO 217Ø
214Ø RESUME 215Ø
215Ø CLOSE #2
216Ø ER%=7:GOSUB 162Ø:GOTO 127Ø
217Ø CLOSE #2:OPEN PN$ FOR APPEND AS #2
218Ø NN%=-1:GOTO 128Ø
```

PLANR (FROM PAGE 39)

95 
96 
97 
98 
99 
100 
101 
102 
103 
104 
105 
106 
107 
108 
109 
110 
111 
112 
113 
114 
115 
116 
117 
118 
119 
120 
121 
122 
123 
124 
125 
126 
127 
128 
129 
130 
131 
132 
133
134 
135
136

137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178

Abort	Skip Line	Stop & Save	Resume	Yes	No

ALABAMA	
Birmingham	Jefferson News Co.
Florence	Anderson News Co.
Madison	Madison Books
ARKANSAS	
Fayetville	Vaughn Electronics/Radio Shack
ARIZONA	
Phoenix	Computer Pro
	Tri-Tek
Tucson	Anderson News Co.
CALIFORNIA	
Citrus Heights	Software Plus
Half Moon Bay	Strawflower Electronics
Livermore	Software Galeria
North Hollywood	Levity Distributors
Santa Rosa	Sawyer's News
Sunnyvale	Computer Literacy
DELAWARE	
Middletown	Delmar Co.
Wilmington	Normar Inc.—The Smoke Shop
FLORIDA	
Boca Raton	Software, Software, Inc.
Ft. Lauderdale	Software Connection
N. Miami Beach	Almar Bookstore
Panama City	Computer Systems Group
Pensacola	Anderson News Co.
Tallahassee	Anderson News Co.
GEORGIA	
Atlanta	Guild News Agency
ILLINOIS	
Chicago	Chas. Levy Distr.
	Kroch's & Brentano's Jackson Street
	Kroch's & Brentano's Wabash Ave.
	Prairie News Agency
Oakbrook	Kroch's & Brentano's
INDIANA	
Scottsburg	Radio Shack of Scottsburg
IOWA	
Ames	Myers TV & Stereo Repair
KANSAS	
Wichita	Lloyd's Radio
KENTUCKY	
Louisville	Hawley-Cooke Bookstores
Prospect	Falsoft, Inc.
LOUISIANA	
Gretna	Computer Supply Store
Slidell	Radio Shack
MASSACHUSETTS	
Littleton	Computer Plus
Marlboro	Radio Shack
MICHIGAN	
Petosky	B & L Sound, Inc.
Rochester	Rochester Book Center
Sterling Heights	Software City
	Programs, Inc.
Wyoming	Gerry's Book Co.
MISSOURI	
Independence	Digital Enterprises, Inc.
St. Louis	Software Center
NEBRASKA	
Lincoln	Hobby Town
NEW HAMPSHIRE	
West Lebanon	Verham News Corp.
NEW JERSEY	
Clinton	Micro World II
Hackensack	Total Circulation Service
Pennsville	Dave's Electronic Radio Shack
NEW MEXICO	
Alamogardo	New Horizons Computer Systems
Albuquerque	News and Bookstore
	Page One Newsstand
NEW YORK	
Hudson Falls	G.A. West & Co.
NORTH CAROLINA	
Cary	News Center in Cary Village
Sylva	Total Data Systems
NORTH DAKOTA	
Fargo	Computer Associates
OHIO	
Cincinnati	Cinsoft
Toledo	Leo's
OKLAHOMA	
Oklahoma City	Merit Computers
RHODE ISLAND	
Newport	Kelly's Variety
TENNESSEE	
Knoxville	Anderson News Co.
Memphis	Computer Center
Nashville	Mosko's Book Store
Smyrna	Delker Electronics, Inc.
TEXAS	
Ft. Worth	A & A International
VIRGINIA	
Alexandria	Alonso Book & Period
Norfolk	I-O Computer
WASHINGTON	
Kennewick	Satellite TV Specialists
WISCONSIN	
Ladysmith	Electronics, Etc.
CANADA:	
ALBERTA	
Calgary	Rainbow Software Services
Edmonton	CMD Micro
	Kelly Software Distributors
BRITISH COLUMBIA	
Sidney	Sidney Electronics
QUEBEC	
LaSalle	Messageries De Presse Benjamin
NOVA SCOTIA	
Halifax	Atlantic News

Also available at selected B.Dalton Bookseller stores

Advertisers Index

PCM Advertising and Marketing Representatives

Cindy Shackleford
Director, West Coast Office
12110 Meridian South — Suite 8
P.O. Box 73-578
Puyallup, WA 98373-0578
(206) 848-7766

Kim Vincent
Advertising Representative
The Falsoft Building
P.O. Box 385
Prospect, KY 40059
(502) 228-4492

Join us for our very first show . . .

PCMfest℠

Princeton, N.J. *Oct. 11-13*

You're invited to PCMfest, a great new show that's just for your Tandy computer. Sponsored by PCM, *The Personal Computing Magazine for Tandy Computer Users*, it's a wonderful way to meet and exchange information with those who share your interest in the new generation of Tandy computers.

You'll also discover the greatest variety of products ever for your computer because all of the favorites and a lot of new ones will be featured in the exhibits of PCM advertisers. Try out that new program and take it home that very day!

Along with other PCM readers, you'll meet the top national experts on your computer, including those who write for or who are written about in PCM. They will answer your questions on the spot.

PCMfest also will include a comprehensive lineup of free seminars on topics of immediate concern — and all of them designed to help you get the most out of your Tandy computer.

The Hyatt Regency—Princeton will be offering special rates ($65, single or double room) for PCMfest. The show opens Friday evening with a 7 p.m. to 10 p.m. session. It's a daytime-only show Saturday — the exhibits open at 10 a.m. and run continuously until 6 p.m. On Sunday, the exhibit hall opens at 11 a.m. and closes at 4 p.m.

Tickets may be obtained directly from PCM. We'll also send you a special reservation form so you can take advantage of the special room rate. Come to PCMfest and let's celebrate the new generation of Tandy computers!

RAINBOWfest

Your admission to PCMfest also entitles you to attend RAINBOWfest, the highly popular show for the Tandy Color Computer which will run concurrently with PCMfest at the same location.

Keynote Speaker

Bill Barden, respected author of 30 books on various computer subjects, will speak at our Saturday Breakfast at 8 a.m. (Separate tickets required.)

Free Seminars

Sam Redmon — *The Future of Portable Computing*
As co-founder of the Dallas-based Portable Computer Support Group, Sam markets a wide variety of software and hardware for the Tandy portables.

Harry Brawley — *The Portable Computer Environment*
President of Sigea Systems, he is creator of the Telecommuter integrated communications and word processing software.

Carl Oppedahl — *Computer Law and Portable Computers*
An attorney practicing law in Manhattan, Carl specializes in technological litigation. He is the author of a newly-published book titled *Inside the TRS-80 Model 100.*

Bill Barden — *Computer Languages*
Discussion will include choosing the right language for your application and an overview of the most popular microcomputer languages including BASIC, C, PASCAL and Assembly.

Bob Covington — *Data Communications*
A writer and editor from St. Louis, Bob is author of "The Portable Machine" series for the Tandy 100 and the "Subroutine City" series for Tandy's MS-DOS computers.

Howard Wolowitz — *Database Management*
President of Small Computer Company, Howard is one of the developers of Small's popular *Profile* and *filePro* database management systems.

Danny Humphress — *ViaNet—Tandy's New Intra-Office Communications System*
PCM managing editor and author of the "MS-DOSsier" and "dBASE Tutor" series, Danny also owns a software firm in Louisville.

YES, I'm coming to Princeton! I want to save by buying tickets now at the special advance sale price.

Please send me:

_____ three-day tickets at $9 each total _____

_____ one-day tickets at $7 each total _____
(Circle one) Friday Saturday Sunday

_____ Saturday Breakfast at $12 each total _____

Handling Charge $1 _____

TOTAL ENCLOSED _____

U.S. FUNDS ONLY, PLEASE

☐ Also send me a hotel reservation card for the Hyatt Regency-Princeton ($65, single or double room).

Name (please print) _____

Address _____

City _____ State _____

Telephone _____ ZIP _____

Company _____

☐ Payment enclosed, OR ☐ Charge to:
☐ VISA ☐ MasterCard ☐ American Express

Account Number _____

Exp. Date _____

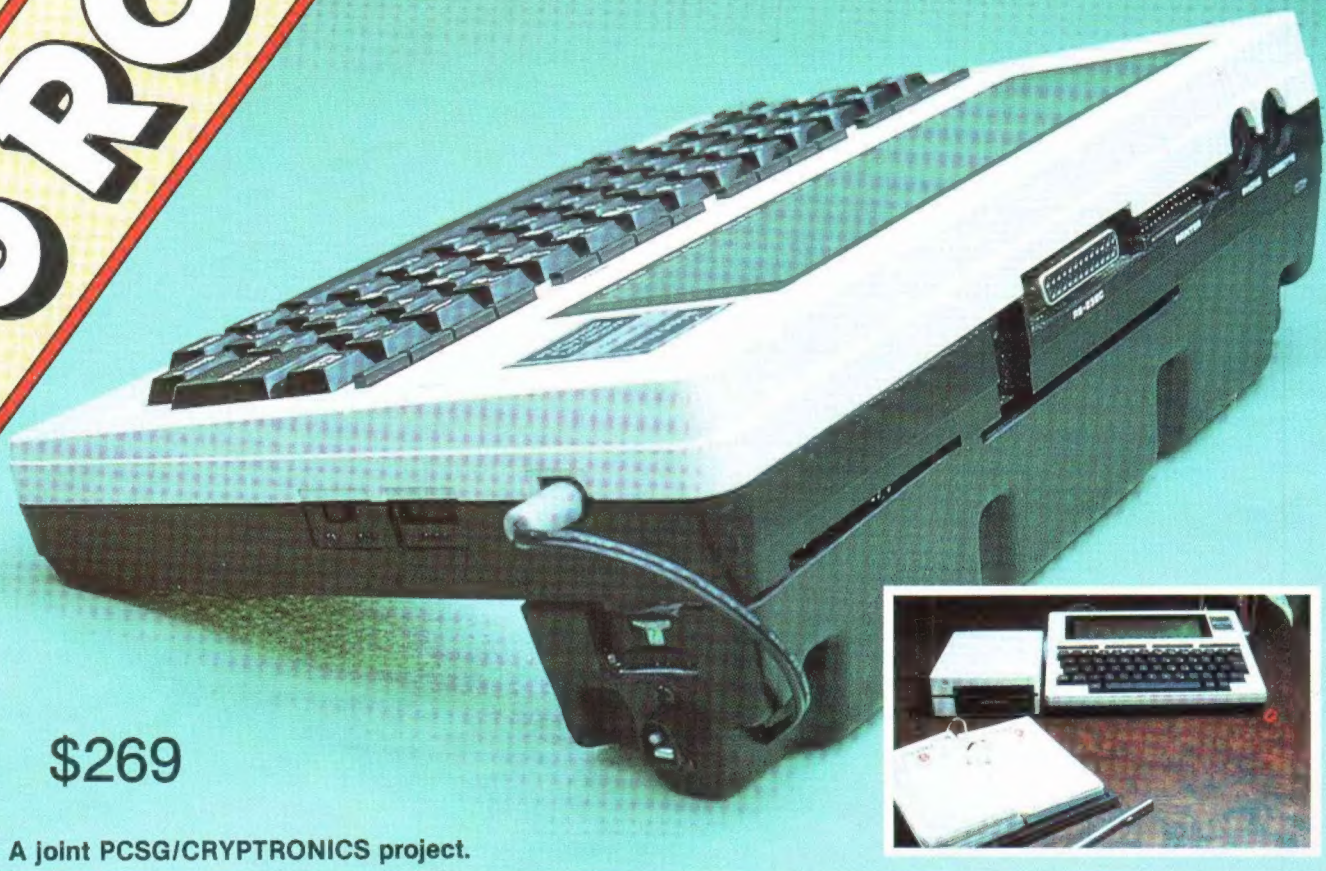
Signature _____

Make checks payable to: PCM. Mail to: PCMfest, the Falsoft Building, P.O. Box 385, Prospect, KY 40059. To make reservations by phone, call: (502) 228-4492.

Advance ticket deadline: October 4, 1985. Orders received less than two weeks prior to show opening will be held for you at the door. Tickets will also be available at the door at a slightly higher price.

6 ROM BANK

plus powerful Nicad battery pack that gives 30 hours of power to your Model 100



$269

A joint PCSG/CRYPTRONICS project.

Now you can access LUCID, DISK+, WRITE ROM, RAM+ and others instantly.

The 6 ROM BANK plus battery pack lets you have the ultimate Model 100 system. When you get your unit just press in up to 6 ROMS into the sockets then push in a simple plug into your Model 100's ROM socket. The ROM bank has two posts on either end that insert into the two little holes on the underside of your Model 100.

The ROM bank props up the Model 100 at the same angle and height as those little legs you've seen. The ROM BANK itself is only about 1½" deep and it runs the width of your Model 100. It only weighs one pound. It not only installs instantly, but it pops free in a second if you need everything to lie flat in a briefcase.

Change from ROM to ROM with the touch of a thumb switch.

You can go from LUCID to WRITE to DISK+ to any other ROMS just by turning the thumb switch at the side of the ROM bank. The 6 ROM BANK is a sturdy well built construction that looks like it is a part of your Model 100.

What is also fantastic is that the ROM bank has a powerful NICAD battery and recharger built right in. This power source supplies up to 30 hours of life to your Model 100 with just 6 hours of recharge. What's nice is that it recharges right from your Model 100's power adapter. This is a quick charge system and if you need power in a hurry, you can get 6 hours of life for your Model 100 by just charging for an hour and half.

Suddenly, the Model 100 is a very powerful computer.

If you have the HOLMES portable disk drive with its powerful bundled database software package or the CRYPTRONICS 128K RAM expansion along with LUCID, WRITE ROM and DISK+ in the 6 ROM bank with its rapid rechargable NICAD power source, you have the ultimate portable system. It's all available only from PCSG.

As usual we don't want you to take our word for it. The 6 ROM BANK is sold on a 30 day trial. If you are not satisfied simply return within thirty days for a full refund. MC, VISA, AM. EX. or C.O.D.

1-214-351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd. No. 207, Dallas, Tx. 75229

VISA / MASTER CARD / AM. EXP. / C.O.D.